DOCUMENT RESUME

ED 477 468 IR 058 321

AUTHOR	Voorhees, Ellen M., Ed.; Harman, Donna K., Ed.
TITLE	The Text REtrieval Conference (TREC-9) (9th, Gaithersburg, Maryland, November 13-16, 2000). NIST Special Publication.
INSTITUTION	National Inst. of Standards and Technology, Gaithersburg, MD.; Advanced Research Projects Agency (DOD), Washington, DC.
REPORT NO	NIST-Pub-500-249
PUB DATE	2000-11-00
NOTE	819p.; Conference was also sponsored by Advanced Research and Development Activity (ARDA).
AVAILABLE FROM	For full text: http://trec.nist.gov/pubs/trec9/t9_proceedings.html.
PUB TYPE	Collected Works - Proceedings (021)
EDRS PRICE	EDRS Price MF05/PC33 Plus Postage.
DESCRIPTORS	Conferences; Foreign Countries; *Information Retrieval; *Information Systems; Information Technology; Interaction; Workshops; World Wide Web
IDENTIFIERS	Filters; Query Processing; *Question Answering; *Text Searching; United States

ABSTRACT

This report constitutes the proceedings of the ninth Text REtrieval Conference (TREC-9). The conference was co-sponsored by the National Institute of Standards and Technology (NIST), the Defense Advanced Research Projects Agency (DARPA), and the Advanced Research and Development Agency (ARDA). Approximately 175 people attended the conference, including representatives from 17 countries. The conference was the ninth in an ongoing series of workshops to evaluate new technologies for text retrieval and related information-seeking tasks. The seven tracks included in TREC-9 were Web retrieval, cross-language retrieval, spoken document retrieval, query analysis, question answering, interactive retrieval, and filtering. A total of 69 groups submitted retrieval results to one or more of the workshop's tracks. The workshop included paper sessions and discussion groups. This proceedings includes papers from most of the participants, track reports that define the problem addressed by the track plus summarize the main track results, and tables of individual group results. (Author/MES)

Reproductions supplied by EDRS are the best that can be made
from the original document.

ED 477 468

The Text REtrieval Conference (TREC-9)
(9th, Gaithersburg, Maryland, November 13-16, 2000)
NIST Special Publication

Edited by

Ellen M. Voorhees and Donna K. Harman

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION
CENTER (ERIC)

☐ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

☐ Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

PERMISSION TO REPRODUCE AND
DISSEMINATE THIS MATERIAL HAS
BEEN GRANTED BY

E. Voorhees

TO THE EDUCATIONAL RESOURCES
INFORMATION CENTER (ERIC)

1

IR 058 321

BEST COPY AVAILABLE

NIST Special Publication XXX-XXX: The Ninth Text REtrieval Conference (TREC 9)

TREC home Publications home Help

NOTE: Portions of these proceedings are available in either PS or PDF formats. Applicable files have been compressed (.gz), in order to preserve disk space. Papers need to be saved and unzipped before viewing in the appropriate software. For additional information on these utilities, please reference the [help] file.

TABLE OF CONTENTS

FOREWORD

ABSTRACT

PAPERS

- *alphabetically, by organization*
- *alphabetically, by track*

---

1. **Overview of the Ninth Text REtrieval Conference (TREC-9)**, *page 1*
   E. Voorhees, D. Harman, National Institute of Standards and Technology
2. **TREC-9 Cross-Language Information Retrieval (English-Chinese) Overview**, *page 15*
   F. Gey, A. Chen, University of California, Berkeley
3. **The TREC-9 Filtering Track Final Report**, *page 25*
   S. Robertson, Microsoft Research
   D.A. Hull, WhizzBang Labs

The TREC-9 Interactive Track Report, *page 41*
W. Hersh, Oregon Health Sciences University
Paul Over, National Institute of Standards and Technology

4. **Query Expansion Seen Through Return Order of Relevant Documents**, *page 51*
W. Liggett, NIST
C. Buckley, SabIR Research, Inc.

5. **Overview of the TREC-9 Question Answering Track**, *page 71*
E. Voorhees, NIST

6. **The TREC-9 Query Track**, *page 81*
Chris Buckley, Sabir Research, Inc.

7. Spoken Document Retrieval Track Slides
J. Garofolo, J. Lard, E. Voorhees, NIST

8. **Overview of the TREC-9 Web Track**, *page 87*
D. Hawking, CSIRO Mathematical and Information Sciences

9. **Structuring and Expanding Queries in the Probabilistic Model**, *page 427*
Y. Ogawa, H. Mano, M. Narita, S. Honma, RICOH Co., Ltd.

10. **IIT TREC-9 - Entity Based Feedback with Fusion**, *page 241*
A. Chowdhury, S. Beitzel, E. Jensen, M. Sai-lee, D. Grossman,
O. Frieder, Illinois Institute of Technology
M.C. McCabe, U.S. Government
D. Holmes, NCR Corporation

11. **TREC-9 Cross Language, Web and Question-Answering Track Experiments using PIRCS**, *page 419*
K.L. Kwok, L. Grunfeld, N. Dinstl, M. Chan, Queens College, CUNY

12. **TREC-9 CLIR Experiments at MSRCN**, *page 343*
J. Gao, E. Xun, M. Zhou, C. Huang, Microsoft Research China
J-Y Nie, Université de Montréal
J. Zhang, Y. Su, Tsinghua University China

13. **FALCON: Boosting Knowledge for Answer Engines**, *page 479*
S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea,
M. Surdeanu, R. Bunescu, R. Gîrju, V. Rus, P. Morarescu, Southern Methodist University

14. **IBM's Statistical Question Answering System**, *page 229*
A. Ittycheriah, M. Franz, W-J Zhu,

A. Ratnaparkhi, IBM T.J. Watson Research Center
R.J. Mammone, Rutgers University

15. **Question Answering by Passage Selection (MultiText Experiments for TREC-9)**, *page 673*
C.L.A. Clarke, G.V. Cormack, D.I.E. Kisman, T.R. Lynam, University of Waterloo

16. **Question Answering in Webclopedia**, *page 655*
E. Hovy, L. Gerber, U. Hermjakob, M. Junk, C-Y Lin, University of Southern California

17. **Filters and Answers: The University of Iowa TREC-9 Results**, *page 533*
E. Catona, D. Eichmann, P. Srinivasan, University of Iowa

18. **The LIMSI SDR System for TREC-9**, *page 335*
J.-L. Gauvain, L. Lamel, C. Barras, G. Adda, Y. de Kercardio, LIMSI-CNRS

19. **Microsoft Cambridge at TREC-9: Filtering Track**, *page 361*
S.E. Robertson, S. Walker, Microsoft Research Ltd., UK

20. **AT&T at TREC-9** , *page 103*
A. Singhal, M. Kaszkiel, AT&T Labs-Research

21. **Spoken Document Retrieval for TREC-9 at Cambridge University**, *page 117*
S.E. Johnson, P. Jourlin, K. Spärck Jones, P.C. Woodland, Cambridge University

22. **kNN at TREC-9**, *page 127*
T. Ault, Y. Yang, Carnegie Mellon University

23. **YFilter at TREC-9**, *page 135*
Y. Zhang, J. Callan, Carnegie Mellon University

24. **Passive Feedback Collection--An Attempt to Debunk the Myth of Clickthroughs**, *page 141*
C. Vogt, Chapman University

25. **TREC-9 CLIR at CUHK: Disambiguation by Similarity Values Between Adjacent Words**, *page 151*
H. Jin, K-F Wong, The Chinese University of Hong Kong

26. **Syntactic Clues and Lexical Resources in Question-Answering**, *page 157*
K.C. Litkowski, CL Research

27. **Dublin City University Experiments in Connectivity Analysis for TREC-9**, *page 179*
C. Gurrin, A.F. Smeaton, Dublin City University

28. **FDU at TREC-9: CLIR, Filtering and QA Tasks**, *page 189*
L. Wu, X-j Huang, Y. Guo, B. Liu, Y. Zhang, Fudan University

29. **Fujitsu Laboratories TREC-9 Report**, *page 203*
I. Namba, Fujitsu Laboratories, Ltd.

30. **Hummingbird's Fulcrum SearchServer at TREC-9**, *page 209*
S. Tomlinson, T. Blackwell, Hummingbird

31. **English-Chinese Information Retrieval at IBM**, *page 223*
M. Franz, J.S. McCarley, W-J Zhu, IBM T.J. Watson Research Center

32. **One Search Engine or Two for Question-Answering**, *page 235*
J. Prager, E. Brown, IBM T.J. Watson Research Center
D.R. Radev, University of Michigan
K. Czuba, Carnegie-Mellon University

33. **Training Context-Sensitive Neural Networks with Few Relevant Examples for the TREC-9 Routing**, *page 257*
M. Stricker, Informatique-CDC and ESPCI
F. Vichot, F. Wolinski, Informatique-CDC
G. Dreyfus, ESPCI

34. **Mercure at trec9: Web and Filtering tasks**, *page 263*
M. Abchiche, M. Boughanem, T. Dkaki, J. Mothe, C. Soule Dupuy,
M. Tmar, IRIT-SIG

35. **The HAIRCUT System at TREC-9**, *page 273*
P. McNamee, J. Mayfield, C. Piatko, The Johns Hopkins University, APL

36. **Experiments on the TREC-9 Filtering Track**, *page 295*
K. Hoashi, K. Matsumoto, N. Inoue, K. Hashimoto,
KDD R&D Laboratories, Inc.
T. Hasegawa, K. Shirai, Waseda University

37. **TREC-9 Experiments at KAIST: QA, CLIR and Batch Filtering**, *page 303*
K-S Lee, J-H Oh, JX Huang, J-H Kim, K-S Choi, Korea Advanced Institute of Science and Technology

38. **Question Answering Considering Semantic Categories and Co-Occurrence Density**, *page 317*
S-M Kim, D-H Baek, S-B Kim, H-C Rim, Korea University

39. **QALC--The Question-Answering System of LIMSI-CNRS**, *page 235*
O. Ferret, B. Grau, M. Hurault-Plantet, G. Illouz, C. Jacquemin, LIMSI-CNRS
N. Masson, P. Lecuyer, Bertin Technologies

40. **Question Answering Using a Large NLP System**, *page 355*
D. Elworthy, Microsoft Research Ltd.

41. **NTT DATA TREC-9 Question Answering Track Report,** *page 399*
T. Takaki, NTT Data Corporation

42. **Description of NTU QA and CLIR Systems in TREC-9**, *page 389*
C-J Lin, W-C Lin, H-H Chen, National Taiwan University

43. **Further Analysis of Whether Batch and User Evaluations Give the Same Results with a Question-Answering Task**, *page 407*
W. Hersh, A. Turpin, L. Sacherek, D. Olson, S. Price, B. Chan,
D. Kraemer, Oregon Health Sciences University

44. **Melbourne TREC-9 Experiments**, *page 437*
D. D'Souza, M. Fuller, J. Thom, P. Vines, J. Zobel, RMIT University
O. de Kretser, University of Melbourne
R. Wilkinson, M. Wu, CSIRO, Division of Mathematics and Information Science

45. **Support for Question-Answering in Interactive Information Retrieval: Rutgers' TREC-9 Interactive Track Experience**, *page 463*
N.J. Belkin, A. Keller, D. Kelly, J. Perez-Carballo, C. Sikora, Y. Sun, Rutgers University

46. **Halfway to Question Answering**, *page 489*
W.A. Woods, S. Green, P. Martin, A. Houston, Sun Microsystems Laboratories

47. **Question Answering: CNLP at the TREC-9 Question Answering Track**, *page 501*
A. Diekema, X. Liu, J. Chen, H. Wang, N. McCracken, O. Yilmazel, E.D. Liddy, Syracuse University, School of Information Studies

48. **CINDOR TREC-9 English-Chinese Evaluation**, *page 379*
M.E. Ruiz, S. Rowe, M. Forrester, P. Sheridan, MNIS-TextWise Labs

49. **TNO-UT at TREC-9: How Different are Web Documents?**, *page 665*
W. Kraaij, TNO-TPD
T. Westerveld, University of Twente, CTIT

50. **A Semantic Approach to Question Answering Systems,** *page 511*
J.L. Vicedo, A. Ferrandez, Universidad de Alicante

51. **The PISAB Question Answering System**, *page 621*
G. Attardi, C. Burrini, Universitá di Pisa - Italy

52. **Goal-Driven Answer Extraction**, *page 563*
M. Laszlo, L. Kosseim, G. Lapalme, Université de Montréal

53. **The System RELIEFS: A New Approach for Information Filtering**, *page 573*
C. Brouard and J-Y Nie, Université de Montréal

54. **Report on the TREC-9 Experiment: Link-based Retrieval and Distributed**

Collections, *page 579*
J. Savoy, Y. Rasolofo, Université de Neuchâtel

55. **English-Chinese Cross-Language IR Using Bilingual Dictionaries**, *page 517*
A. Chen, H. Jiang, School of Information Management and Systems, University of California at Berkeley
F. Gey, UC Data Archive & Technical Assistance (UC DATA), University of California at Berkeley

56. **Question Answering, Relevance Feedback and Summarisation: TREC-9 Interactive Track Report**, *page 523*
N. Alexander, C. Brown, J. Jose, I. Ruthven, A. Tombros, University of Glasgow

57. **INQUERY and TREC-9**, *page 551*
J. Allan, M.E. Connell, W.B. Croft, F-F Feng, D. Fisher, X. Li, Center for Intelligent Information Retrieval, Department of Computer Science, University of Massachusetts

58. **Information Space Based on HTML Structure**, *page 601*
G. Newby, University of North Carolina, Chapel Hill

59. **Web Document Retrieval Using Passage Retrieval, Connectivity Information, and Automatic Link Weighting--TREC-9 Report**, *page 611*
F. Crivellari, M. Melucci, University of Padova (Italy)

60. **The Thisl SDR System at TREC-9**, *page 627*
S. Renals, D. Abberley, University of Sheffield, UK

61. **University of Sheffield TREC-9 Q&A System**, *page 635*
S. Scott, R. Gaizauskas, University of Sheffield

62. **The Mirror DBMS at TREC-9**, *page 171*
A.P. de Vries, CWI, Amsterdam, The Netherlands

63. **TREC-9 Cross-lingual Retrieval at BBN**, *page 106*
J. Xu, R. Weischedel, BBN Technologies

64. **Sheffield Interactive Experiment at TREC-9**, *page 645*
M. Beaulieu, H. Fowkes, H. Joho, University of Sheffield, UK

65. **Reflections on "Aboutness" TREC-9 Evaluation Experiments at Justsystem**, *page 281*
S. Fujita, Justsystem Corporation

66. **Incrementality, Half-life, and Threshold Optimization for Adaptive Document Filtering**, *page 589*
A. Arampatzis, J. Beney, C.H.A. Koster, T.P. van der Weide, University of Nijmegen

67. **ACSys/CSIRO TREC-9 Experiments**, *page 167*
D. Hawking, CSIRO Mathematics and Information Sciences

68. **Kasetsart University TREC-9 Experiments**, *page 289*
P. Narasetsathaporn, A. Rungsawang, Kasetsart University, Bangkok, Thailand

69. **SabIR Research at TREC-9**, *page 475*
C. Buckley, J. Walz, SabIR Research

70. **A Simple Question Answering System**, *page 249*
R. J. Cooper, S.M. Rüger, Imperial College of Science, Technology and Medicine

71. **TREC-9 Experiments at Maryland: Interactive CLIR**, *page 543*
D.W. Oard, G-A Levow, C.I. Cabezas, University of Maryland

72. **Logical Analysis of Data in the TREC-9 Filtering Track**, *page 453*
E. Boros, P.B. Kantor, D.J. Neu, Rutgers University

73. **Another Sys Called Qanda**, *page 369*
E. Breck, J. Burger, L. Ferro, W. Greiff, M. Light, I. Mani, J. Rennie, The MITRE Corporation

APPENDICES

A. **TREC-9 Results**, *page A-1*
   - **Evaluation Techniques and Measures**, *page A-15*
   - **Cross-Language Runs List**, *page A-2*
   - **Cross-Language track results**, *page A-20*
   - **Filtering Runs List**, *page A-3*
   - **Filtering track results**, *page A-58*
   - **Interactive track results**, *page A-88*
   - **Question Answering Runs List**, *page A-5*
   - **Question Answer track results**, *page A-100*
   - **Query track results**, *page A-95*
   - **Spoken Document Retrieval Runs List**, *page A-12*
   - **Spoken Document Retrieval track results**, *page A-178*
   - **Web Runs List**, *page A-12*
   - **Web track results**, *page A-210*

TREC-9 System Descriptions

Last updated: Thursday, 11-Oct-01 15:54:35
Date created: Tuesday, 01-Aug-00
trec@nist.gov

Foreword

This report constitutes the proceedings of the ninth Text REtrieval Conference (TREC-9) held in Gaithersburg, Maryland, November 13–16, 2000. The conference was co-sponsored by the National Institute of Standards and Technology (NIST), the Defense Advanced Research Projects Agency (DARPA), and the Advanced Research and Development Agency (ARDA). Approximately 175 people attended the conference, including representatives from seventeen different countries. The conference was the ninth in an on-going series of workshops to evaluate new technologies for text retrieval and related information-seeking tasks. Sixty-nine groups submitted retrieval results to one or more of the workshop's tracks.

The workshop included plenary sessions, discussion groups, a poster session, and demonstrations. Because the participants in the workshop drew on their personal experiences, they sometimes cited specific vendors and commercial products. The inclusion or omission of a particular company or product implies neither endorsement nor criticism by NIST. Any opinions, findings, and conclusions or recommendations expressed in the individual papers are the authors' own and do not necessarily relect those of the sponsors.

The sponsorship of the U.S. Department of Defense is gratefully acknowledged, as is the tremendous work of the program committee and the track coordinators.

Ellen Voorhees,
Donna Harman
August 29, 2001

TREC-9 Program Committee

Ellen Voorhees, NIST, chair
James Allan, University of Massachusetts at Amherst
Nick Belkin, Rutgers University
Chris Buckley, Sabir Research, Inc.
Jamie Callan, Carnegie Mellon University
Susan Dumais, Microsoft
Donna Harman, NIST
David Hawking, CSIRO
Bill Hersh, Oregon Health Sciences Institute
Darryl Howard, U.S. Department of Defense
David Hull, WhizzBang Labs
John Prange, U.S. Department of Defense
Steve Robertson, Microsoft
Amit Singhal, AT&T Labs–Research
Karen Sparck Jones, University of Cambridge, UK
Tomek Strzalkowski, State University of New York, Albany
Ross Wilkinson, CSIRO

BEST COPY AVAILABLE

Abstract

This report constitutes the proceedings of the ninth Text REtrieval Conference (TREC-9) held in Gaithersburg, Maryland, November 13–16, 2000. The conference was co-sponsored by the National Institute of Standards and Technology (NIST), the Defense Advanced Research Projects Agency (DARPA), and the Advanced Research and Development Agency (ARDA). Sixty-nine groups including participants from seventeen different countries were represented.

TREC-9 is the latest in a series of workshops designed to foster research in text retrieval and related technologies. The previous eight TRECs each had an "ad hoc" main task through which eight large test collections were built. In recognition that sufficient infrastructure exists to support researchers interested in this traditional retrieval task, the ad hoc main task was discontinued in TREC-9 so that more TREC resources could be focused on building evaluation infrastructure for other retrieval tasks (called "tracks"). The seven tracks included in TREC-9 were web retrieval, cross-language retrieval, spoken document retrieval, query analysis, question answering, interactive retrieval, and filtering.

The conference included paper sessions and discussion groups. This proceedings includes papers from most of the participants (some groups did not submit papers), track reports that define the problem addressed by the track plus summarize the main track results, and tables of individual group results. The TREC-9 proceedings web site also contains system descriptions that detail the timing and storage requirements of the different runs.

BEST COPY AVAILABLE

Alphabetical Index of TREC-9 Papers by Organization

TREC home Publications home Help

AT&T Labs-Research

AT&T at TREC-9, *page 103*

BBN Technologies

TREC-9 Cross Lingual Retrieval at BBN, *page 106*

Bertin Technologies

QALC--The Question-Answering System of LIMSI-CNRS, *page 325*

Cambridge University

Spoken Document Retrieval for TREC-9 at Cambridge University, *page 117*

Carnegie Mellon University

kNN at TREC-9, *page 127*

YFilter at TREC-9, *page 135*

One Search Engine or Two for Question-Answering, *page 235*

Chapman University

Passive Feedback Collection--An Attempt to Debunk the Myth of Clickthroughs, *page 141*

Chinese University at Hong Kong

TREC-9 CLIR at CUHK Disambiguation by Similarity Values Between Adjacent Words, *page 151*

CL Research

Syntactic Clues and Lexical Resources in Question-Answering, *page 157*

CSIRO Mathematics and Information Sciences

ACSys/CSIRO TREC-9 Experiments, *page 167*

Melbourne TREC-9 Experiments, *page 437*

Overview of the TREC-9 Web Track, *page 87*

CWI, Amsterdam

The Mirror DBMS at TREC-9, *page 171*

Dublin City University

Dublin City University Experiments in Connectivity Analysis for TREC-9, *page 179*

ESPCI

Training Context-Sensitive Neural Networks with Few Relevant Examples for the TREC-9 Routing , *page 257*

Fudan University

FDU at TREC-9: CLIR, Filtering and QA Tasks, *page 189*

Fujitsu Laboratories, Ltd.

Fujitsu Laboratories TREC-9 Report, *page 203*

Hummingbird

Hummingbird's Fulcrum SearchServer at TREC-9, *page 209*

IBM T.J. Watson Research Center

English-Chinese Information Retrieval at IBM, *page 223*

One Search Engine or Two for Question-Answering, *page 235*

IBM's Statistical Question Answering System, *page 229*

Illinois Institute of Technology

IIT TREC-9-Entity Based Feedback with Fusion, *page 241*

Imperial College of Science, Technology and Medicine

A Simple Question Answering System, *page 249*

Informatique-CDC

Training Context-Sensitive Neural Networks with Few Relevant Examples for the TREC-9 Routing, *page 257*

IRIT-SIG

Mercure at trec9: Web and Filtering tasks, *page 263*

Johns Hopkins University, APL

The HAIRCUT System at TREC-9, *page 273*

Justsystem Corporation

Reflections on "Aboutness" TREC-9 Evaluation Experiments at Justsystem, *page 281*

KDD R&D Laboratories, Inc.

Experiments on the TREC-9 Filtering Track, *page 295*

Korea Advanced Institute of Science and Technology

TREC-9 Experiments at KAIST: QA, CLIR and Batch Filtering, *page 303*

Korea University

Question Answering Considering Semantic Categories and Co-Occurence Density, *page 317*

LIMSI-CNRS

QALC--The Question-Answering System of LIMSI-CNRS, *page 325*

The LIMSI SDR System for TREC-9, *page 335*

Microsoft Research

The TREC-9 Filtering Track Final Report, *page 25*

Microsoft Research China

TREC-9 CLIR Experiments at MSRCN, *page 343*

Microsoft Research Ltd.

Question Answering Using a Large NLP System, *page 355*

Microsoft Research Ltd., UK

Microsoft Cambridge at TREC-9: Filtering Track, *page 361*

The MITRE Corporation

Another Sys Called Qanda, *page 369*

MNIS-TextWise Labs

CINDOR Trec-9 English-Chinese Evaluation, *page 379*

National Institute of Standards and Technology

Overview of the Ninth Text REtrieval Conference (TREC-9), *page 1*

TREC-9 Interactive Track Report, *page 41*

Overview of the TREC-9 Question Answering Track, *page 71*

Spoken Document Retrieval Track Slides

National Taiwan University

Description of NTU QA and CLIR Systems in TREC-9, *page 389*

NCR Corporation

IIT TREC-9-Entity Based Feedback with Fusion, *page 241*

NTT Data Corporation

NTT DATA TREC-9 Question Answering Track Report, *page 399*

Oregon Health Sciences University

Further Analysis of Whether Batch and User Evaluations Give the Same Results with a Question-Answering Task, *page 407*

TREC-9 Interactive Track Report, *page 41*

Queens College, CUNY

TREC-9 Cross Language, Web and Question-Answering Track Experiments using PIRCS, *page 417*

RICOH Co., Ltd.

Structuring and Expanding Queries in the Probablistic Model, *page 427*

RMIT University

Melbourne TREC-9 Experiments, *page 437*

Rutgers University

Support for Question-Answering in Interactive Information Retrieval: Rutgers' TREC-9 Interactive Track Experience, *page 463*

- Logical Analysis of Data in the TREC-9 Filtering Track, *page 453*
- IBM's Statistical Question Answering System, *page 229*

SabIR Research, Inc.

- SabIR Research at TREC-9, *page 475*
- Query Expansion Seen Through Return Order of Relevant Documents, *page 51*
- The TREC-9 Query Track, *page 81*

Southern Methodist University

- FALCON: Boosting Knowledge for Answer Engines, *page 479*

Sun Microsystems Laboratories

- Halfway to Question Answering, *page 489*

Syracuse University

- Question Answering: CNLP at the TREC-9 Question Answering Track, *page 501*

TNO-TPD

- TNO-UT at TREC-9: How Different are Web Documents?, *page 665*

Tsinghua University China

- TREC-9 CLIR Experiments at MSRCN, *page 343*

U.S. Government

- IIT TREC-9 - Entity Based Feedback with Fusion, *page 241*

Universidad de Alicante

- A Semantic Approach to Question Answering Systems, *page 511*

Universitá di Pisa - Italy

- The PISAB Question Answering System, *page 621*

Université de Montréal

- TREC-9 CLIR Experiments at MSRCN, *page 343*
- Goal-Driven Answer Extraction, *page 563*

The System RELIEFS: A New Approach for Information Filtering, *page 573*

Université de Neuchâtel

Report on the TREC-9 Experiment: Link-based Retrieval an Distributed Collections, *page 579*

University of California at Berkeley

TREC-9 Cross-Language Information Retrieval (English-Chinese) Overview, *page 15*

English-Chinese Cross-Language IR Using Bilingual Dictionaries, *page 517*

University of Glasgow

Question Answering, Relevance Feedback and Summarisation: Trec-9 Interactive Track Report, *page 523*

University of Iowa

Filters and Answers: The University of Iowa TREC-9 Results, *page 533*

University of Maryland

TREC-9 Experiments at Maryland: Interactive CLIR, *page 543*

University of Massachusetts

INQUERY and TREC-9, *page 551*

University of Melbourne

Melbourne TREC-9 Experiments, *page 437*

University of Michigan

One Search Engine or Two for Question-Answering, *page 235*

University of Nijmegen

Incrementality, Half-life, and Threshold Optimization for Adaptive Document Filtering, *page 589*

University of North Carolina, Chapel Hill

Information Space Based on HTML Structure, *page 601*

University of Padova, Italy

Web Document Retrieval Using Passage Retrieval, Connectivity Information, and Automatic Link Weighting--TREC-9 Report, *page 611*

University of Sheffield

University of Sheffield TREC-9 QA System, *page 635*
The Thisl SDR System at TREC-9, *page 627*
Sheffield Interactive Experiment at TREC-9, *page 645*

University of Southern California

Question Answering in Webclopedia, *page 655*

University of Twente, CTIT

TNO-UT at TREC-9: How Different are Web Documents?, *page 665*

University of Waterloo, CTIT

Question Answering by Passage Selection (MultiText Experiments for TREC-9), *page 673*

University of Bangkok, Thailand

Kasetsart University TREC-9 Experiments, *page 289*

Waseda University

Experiments on the TREC-9 Filtering Track, *page 295*

WhizzBang Labs

The TREC-9 Filtering Track Final Report, *page 25*

Last updated: Friday, 05-Oct-01 08:50:07
Date created: Tuesday, 01-Aug-00
trec@nist.gov

AT&T at TREC-9

Amit Singhal Marcin Kaszkiel

AT&T Labs–Research
{singhal,martink}@research.att.com

Abstract

This year we come to TREC with a new retrieval system *Tivra* that we have implemented over the last year. Tivra is based on the vector space model, and is mainly designed to do large-scale web search with limited resources. We run Tivra on a cheap Linux box. It currently indexes around 14-15 gigabytes of web data per hour, and allows sub-second web searches for 2-3 word queries on a 700 MHz Pentium box. At the time of submissions Tivra was in its early development stages, and was not fully tested. However, we still submitted runs for both the web tracks – 10 gigabytes and 100 gigabytes. The results look quite reasonable for an untested version of the system. For the 10 gigabytes ad-hoc task, our results are above median for majority of the queries. This is specially notable given that we use only the title portion of the queries whereas the results pool contains results based on both long and short queries. Our results are among the top results in the 100 gigabytes task.

1 Tivra

Over the last year, we have implemented a new retrieval system called Tivra. Tivra is designed to be a large-scale web search engine which runs on relatively inexpensive Linux PCs. This year at TREC, we submitted several runs for the web tracks using an early development version of Tivra.

Tivra maintains a full positional inverted index on the title and the body of a web page. In our last measurements, Tivra indexed about 15 gigabytes of web data in an hour on a 700 MHz Pentium PC using about 1 gigabyte of RAM. We use a short stop-list of 118 words/numbers. At the time of our TREC runs, we did not use any stemming in Tivra. Since then, we have incorporated a plural stemmer into Tivra. We store raw term frequencies, and the corresponding byte-offsets for words. All term weighting is done at query time.

Tivra also builds indices for the anchor texts that point to a web page. In essence, each web page is indexed as three different documents: the page itself, the off-site anchor texts for the page, and the in-site anchor texts for the page. We maintain a distinction between the off-site anchor text (anchor text for in-links coming from outside the web page's site) and the in-site anchor text (anchor text for in-links coming from within the web page's site) to allow more emphasis on in-links from an outside host. The assumption here is that if a page from a different site points to some page, than this in-link is a stronger recommendation for the pointed-to page, as compared to an in-site in-link. The anchor index does not have positional information. This was motivated by the fact that anchor texts are typically short and positional information, which is mainly needed to enforce proximity, is not that important in this case. The total index size is roughly 15% of the raw web data indexed.

At retrieval time, Tivra processes all the inverted lists in document-order. [1] The inverted list are stored sorted by the document-ids. Tivra reads all the inverted list in one go and runs an efficient merge-sort maintaining a heap of top documents. All term weighting is done at this time. Tivra can compute several scores for a document, for example, a score based on off-site anchor texts, a score based on in-site anchor texts, a global tf×idf score, proximity based scores, proximity in title, and so on. These scores can be combined to get the final document score/rank.

BEST COPY AVAILABLE

2 10 Gigabytes Web Task

We submitted six runs for the 10 gigabytes ad-hoc task. Four of them—`att0010gb`, `att0010gbl`, `att0010gbt`, and `att0010gbe`—do not use any linkage information for the documents. The other two runs—`att0010lf` and `att0010glv`—do use anchor text in their ranking. **All runs use only the title portion of the queries.** We strongly believe that very short queries are in the true spirit of current web search engines.

We use *dnb.dtn* scoring scheme developed in our previous TREC work (see [3] for details). Here is a description of our runs:

- `att0010gb`: This run places documents with all query terms ahead of documents which don't have all query terms. If this strict boolean `AND` doesn't get us 1,000 documents, we add high scoring documents that contain at least one of the two most uncommon (as measured by idf) query terms.
- `att0010gbt`: This run is similar to `att0010gb` but it prefers documents with all query terms in the title field ahead of all other documents.
- `att0010gbl`: This run is similar to `att0010gb` but it assigns an extra credit for locality of query terms in the document body. Here locality is implemented as a window of query length (in bytes) + 50 bytes.
- `att0010gbe`: This run is our two pass query expansion run. This is an overly simplified version of our query expansion run described in [3]. Here are the steps:
  - **Pass-1**: Using *dtn* queries and *dnb* documents, a first-pass retrieval is done.
  - **Expansion:** Top ten documents retrieved in the first pass are ***assumed*** to be relevant to the query. Rocchio's method (with parameters $\alpha = 3$, $\beta = 2$, $\gamma = 0$, γ is immaterial since we do not have any non-relevance here) is used to expand the query by adding twenty new words with highest Rocchio weights. [2] To include the *idf*-factor in the expansion process, documents are *dtb* weighted.
  - **Pass-2:** The expanded query is used with *dnb* documents to generate the final ranking of 1,000 documents.
- `att0010glf`: This run incorporates anchor texts for a page into the scoring function. The final document score is:

1.00	×	off-site anchor text based score	+
0.25	×	in-site anchor text based score	+
1.00	×	title based score	+
1.00	×	locality based score	+
1.00	×	global body score	

  We did not have a chance to train these parameters on any data and these are our best guess parameters.
- `att0010glv`: This is a variant of `att0010glf` and in this run the contribution on the anchor text is reduced as the query gets longer. The thought is that short queries benefit from page recommendations by others whereas the long one's don't. Something that the results don't support strongly.

The results are shown in Table 1. As we had expected given the early stages of development of our system, these results are not spectacular but they are definitely reasonable. These results indicate that linkage analysis (in form of anchor text based indexing) doesn't help the retrieval effectiveness. We would not make this claim with certainty in the general web search environment. In all the testing we have done in-house, linkage analysis improves the search precision notably on short queries. It is possible that the results we obtain the TREC environment are in fact an artifact of the environment.

BEST COPY AVAILABLE

Run	Average Precision	Best	>= Median	< Median
att0010gb	0.1341	0	27	23
att0010gbt	0.1182	1	24	25
att0010gbl	0.1380	0	32	18
att0010gbe	0.1464	1	27	22
att0010glf	0.1250	2	26	22
att0010glv	0.1288	0	29	21

Table 1: Results for 10 gigabytes task (title only queries)

Run	P@1	P@5	P@10
att0010gb	0.5357	0.5190	0.4964
att0010glf	0.5476	0.5048	0.4738

Table 2: Results for 100 gigabytes task (88 queries)

3 100 Gigabytes Web Task

We submitted two runs for the large web task—`att00100gb` and `att00100glf`. These runs are just the 100 gigabytes counterpart of the corresponding 10 gigabytes runs. The results in Table 2 are quite impressive given that one of the runs is a plain tf×idf based run. It is quite promising that for over half the queries, the very first document retrieved was judged relevant. Once again we notice that linkage analysis hasn't improved effectiveness. We are still skeptical of this result and are doing a more elaborate test internally.

4 Conclusions

We are pleased by the reasonably good performance of our untested development version of Tivra, our new search engine. Since the official submission we have removed several shortcomings of Tivra and we expect its performance to improve as we test it further. Even though results show that linkage analysis doesn't improve retrieval effectiveness, we are approaching this result with considerable caution. This result can very well be an artifact of the TREC environment. We are currently running a more elaborate experiment in-house to verify this result.

References

[1] M. Kaszkiel, J. Zobel, and R. Sacks-Davis. Efficient passage ranking for document databases. *ACM Transaction on Information Systems*, 17(4):406–439, October 1999.

[2] J.J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System—Experiments in Automatic Document Processing*, pages 313-323, Englewood Cliffs, NJ, 1971. Prentice Hall, Inc.

[3] Amit Singhal, John Choi, Donald Hindle, David Lewis, and Fernando Pereira. AT&T at TREC-7. In E. M. Voorhees and D. K. Harman, editors, *Proceedings of the Seventh Text REtrieval Conference (TREC-6)*, pages 239-252, 1999.

BEST COPY AVAILABLE

TREC-9 Cross-lingual Retrieval at BBN

Jinxi Xu and Ralph Weischedel
BBN Technologies
70 Fawcett Street
Cambridge, MA 02021

1 Introduction

BBN participated only in the cross-language track at TREC-9. We extended the monolingual approach of Miller et al. (1999), which uses hidden Markov models (HMM), by incorporating translation probabilities from Chinese terms to English terms. We will describe our approach in detail in the next section.

This report will explore the following issues:

1. Whether our HMM-based retrieval model is a viable approach to cross-lingual IR. This is answered by its retrieval performance relative to monolingual retrieval performance.
2. The relative contribution of bilingual lexicons and parallel corpora.
3. The impact of query expansion on cross-lingual performance. We will use two types of query expansion: using English terms and Chinese terms.
4. The impact of query length on retrieval performance. We will use three versions of queries: short, which consist of only the title fields, medium, which consist of title and description fields and long, which consist of title, description and narrative fields of the TREC topics.
5. Whether indexing English words in Chinese documents helps cross-lingual IR. Even though the documents in the corpus are in Chinese, many of them also contain some English words. English words in the documents can directly match the query words.
6. Dialect issues. The Chinese language has many dialects. Cantonese, which is used by the TREC-9 corpus, is one example. Since we had lexical resources for Mandarin (standard Chinese) and for Cantonese, we could measure the impact of dialects on cross-lingual IR.

This report includes official results for our submitted runs and results for experimental runs that are designed to help us explore the issues above.

2 A Hidden Markov Retrieval System for Cross-lingual IR

In our approach, the IR system ranks documents by the probability that a Chinese document *D* is relevant given an English query *Q*, *P*(*D is Rel* /*Q*). Using Bayes Rule, and the fact that *P*(*Q*) is constant for a given query, and our initial assumption of a uniform a priori probability that a document is relevant, ranking documents according to *P*(*Q*/*D is Rel*) is the same as ranking them according to *P*(*D is Rel*/*Q*). The approach therefore estimates the probability that a query *Q* is generated, given the document *D* is relevant. A two state Hidden Markov model approximates the query generation process given a document. One state is General English, denoted by GE, in which a term *e* is selected from the English vocabulary. General English words do not describe the content of the document. They are chosen simply because the user is creating a natural language query in English. The other state is the document state *D* in which a Chinese term *c* from the document is selected and translated to an English word *e*.

After a query is generated from a state, the HMM either stays at the current state or transits to the other state to generate the next query term. The process continues until all query terms are produced.

The following parameters specify the model:

1. General English word probabilities $P(e/GE)$, estimated by

   $$P(e/GE) = \text{frequency of } e \text{ in English corpus/size of English corpus.}$$

   Here e is an English word. English news articles in TREC disks 1-5 are used as an English corpus for this purpose.

2. Chinese word probabilities from the document D, $P(c/D)$, estimated by

   $$P(c/D) = \text{frequency of } c \text{ in } D\text{/size of } D$$

   Here c is a Chinese word.

3. Translation probabilities from Chinese words to English words, $P(e/c)$. We assume that translation probabilities are independent of the document. This is not true, but reduces the number of parameters. We used simple translation probabilities from a bilingual lexicon and more sophisticated estimates from parallel texts.

4. Transition probabilities from one state to the other. We assume

   $$P(GE\text{->}D) = P(D\text{->}D) = a \text{ and}$$

   $$P(D\text{->}GE) = P(GE\text{->}GE) = 1\text{-}a.$$

Further we assume a is independent of the document. Using TREC-5/6 queries (Chinese track) as training, we chose a=0.3.

Note we did not use the standard EM (Expectation-Maximization) procedure for parameter estimation, since using EM would require many training queries for each document.

In this model, we estimate the probability of a query given a document as

$$P(Q \mid D \text{ is } rel) = \prod_{e \text{ in } Q} (aP(e \mid GE) + (1-a)P(e \mid D))$$

and

$$P(e \mid D) = \sum_{\text{all Chinese words } c} P(c \mid D)P(e \mid c)$$

Our monolingual retrieval approach is the one proposed by Miller et al (1999). It ranks documents according to:

$$P(Q \mid D \, is \, rel) = \prod_{c \text{ in } Q} (aP(c \mid GC) + (1-a)P(c \mid D))$$

where $P(c/GC)$ is general Chinese probability for word c, which was estimated from the TREC-9 Chinese corpus itself.

3 Lexical Resources

Two manually created bilingual lexicons were used in our experiments:

- one dealing with the Mandarin dialect from the Linguistic Data Consortium (LDC) and
- the CETA lexicon also dealing primarily with Mandarin.

In addition, two parallel corpora were used to generate bilingual lexicons. The parallel corpora are the Hong Kong SAR news (HKNews) and Hong Kong SAR laws (HKLaws), both from LDC. HKNews has around 18,000 pairs of documents in English and Chinese and has 6.3 million English words. HKLaws has 310,000 pairs of sentences in English and Chinese, with 6.6 million English words.

The following steps were taken to use each bilingual lexicon (whether manually generated or automatically derived from parallel copora):

1. Stem Chinese words via a simple algorithm to remove common Chinese suffixes and prefixes (such as "DE" and "BEI").
2. Use the Porter stemmer to stem the English words (Porter, 1980).
3. Split English phrases into words. If an English phrase is a translation for a Chinese word, each word in the phrase is taken as a separate translation for the Chinese word[1].
4. Estimate translation probabilities.

The resulting lexicons consist of a number of English-Chinese word pairs together with translation probabilities.

For those experiments where no parallel corpus was employed, we assumed a uniform distribution on a word's translations. If a Chinese word *c* has *n* translations $e_1, e_2, \ldots e_n,$ each of them will be assigned equal probability, i.e., $P(e_i/c)=1/n$.

For those experiments where a parallel corpus was employed, we used WEAVER to automatically extract additional translation pairs from the parallel corpora to improve the bilingual lexicons. WEAVER is a statistical machine translation toolkit developed by John Lafferty at Carnegie Mellon University. It has a component to automatically derive word translations based on sentence-aligned parallel text. The Chinese texts in the corpora were segmented by BBN's IdentiFinder™, an information extraction system with a built-in segementor. Since the HKNews corpus in its original form was only aligned at the document level, we developed a sentence alignment algorithm to align it at the sentence level. Our algorithm works by performing an initial alignment using a (potentially small) initial bilingual lexicon (the LDC lexicon). A bilingual lexicon was induced from the initial alignment using WEAVER. The induced lexicon supplements the initial lexicon in producing a better alignment, which in turn results in a better lexicon. The process eventually converges and outputs a list of term translations with translation probabilities.

The translations obtained by WEAVER are statistical in nature. In theory, any Chinese term can be translated to any English term with some probability; for the vast majority of word pairs, the probability approaches 0. For each Chinese term, we output up to 20 English terms and discard the rest, in order to keep the size of the lexicon manageable and to save retrieval time. Table 1 shows some statistics about the lexicons used in our experiments.

The lexicon used for our submitted runs is labeled "ALL" in Table 1. It is a combination of all lexical resources described before, LDC, CETA, HKNews and HKLaws. The sets of English-Chinese word pairs in the individual lexicons were unioned and the translation probabilities linearly combined, with coefficients 0.2, 0.4, 0.3 and 0.1 for LDC, CETA, HKNews and HKLaws respectively. The weights were chosen to reflect the value of each lexical source based on the training queries (TREC-5/6 Chinese). To utilize English words in the documents for cross-lingual retrieval, we include an English word as a

---

[1] This is incorrect, but greatly simplified implementation. The correct method would be to treat phrases in the lexicon and in the queries as single tokens. Research in monolingual IR demonstrated that phrase processing is prone to error and does not conclusively improve retrieval performance.

BEST COPY AVAILABLE

translation of itself with probability 1 and add such "translations" to our lexicons. (Such translations are not included in the statistics in Table 1).

Lexicon Name	# of English terms	# of Chinese terms	# of translation pairs
LDC	86,000	137,000	240,000
CETA	35,000	202,000	517,000
HKNews	21,000	75,000	1,266,000
HKLaws	14,000	38,000	543,000
ALL	108,000	371,000	2,470,000

Table 1: Lexicon statistics. All = combination of all four sources

4 Indexing

One problem in indexing Chinese is segmenting the text, since Chinese has no spaces between words. Instead of using a Chinese segmentor, we used a sub-string match algorithm to extract words from a string of Chinese characters. The algorithm examines any sub-string of length 2 or greater and treats it as a Chinese word if it is in our bilingual lexicons. In addition, any single character that is not part of any of the recognized Chinese words in the first step is also treated as a Chinese word. Note that this algorithm can extract a compound Chinese word as well as its components. For example, the Chinese word “LiZhiWuLi” (“particle physics”) as well as the Chinese words “LiZhi” (“particle”) and “WuLi” (“physics”) will be extracted. This seems desirable because it ensures the retrieval algorithm will match both the compound words as well as their components. The reason for using substring match instead of a more sophisticated segmentor is to improve the chance of mapping words in the Chinese document to an English term via the bilingual lexicons. A segmentor may mis-segment (e.g., a segmentation unit may cover the ending of one word and the beginning of another word). It may over-segment (e.g., producing a compound word while the lexicon only defines the components). It may also under-segment (e.g., producing individual words not defined by the lexicon). Substring matching may result in spurious matches, but we believe it is a less serious problem than being unable to map from Chinese to English due to segmentation errors. Of course, Chinese stop words are removed.

5 Query Processing and Query Expansion Issues

Our first step in query processing is to remove stop words from the queries. These include functional words such as “of” and “the” as well as red herrings in TREC topics such as “relevant” and “document”.

Our query expansion procedure works as follows:

1. For each query, retrieve 10 top ranked documents by an initial retrieval
2. Choose 50 expansion terms from the top ranked document. First, terms that only occur in one top ranked document were discarded. Then expansion terms were ranked by their average *tfidf* weight in the top ranked documents. The *tfidf* formula is the one reported in the UMass TREC6 report (Allan et al, 1998). The top 50 terms were added to the query. The expansion terms, as well as the original query terms were “weighted” by the formula

   $$wt(t,Q) = wt_{old}(t,Q) + 4/10 \sum_{1<=i<=10} tfidf(t,d_i)$$

   Q is a query, $wt_{old}(t, Q)$ is the weight of term t in the original query; $tfidf(t, d)$ is the *tfidf* value of t in document d; and d_i's are the retrieved documents. We interpret the "weight" of a query term in the context of our HMM retrieval approach to be the frequency with which

BEST COPY AVAILABLE

the term is generated by the user. Therefore, the weight was used as an exponent in the retrieval function.

We submitted one monolingual run and three cross-lingual runs:

- BBN9MONO: a monolingual run with automatic query expansion. Final queries consist of the original Chinese queries and 50 expansion terms, using the query expansion procedure above.
- BBN9XLC: Cross-lingual without query expansion.
- BBN9XLB: Cross-lingual run with automatic Chinese query expansion. An initial cross-lingual retrieval was performed using the original English queries. Final queries consist of the original English queries and 50 Chinese expansion terms.
- BBN9XLA: Cross-lingual run with English query expansion and Chinese expansion. English terms were selected from top documents retrieved from an English corpus. Then the expanded English queries were run against the Chinese corpus to get 50 Chinese expansion terms. Final queries consist of the original English queries, 50 English expansion terms and 50 Chinese expansion terms.

The English corpus used for query expansion consists of news articles from TREC disks 1-5 (AP, WSJ, SJMN, FT, L. A. TIMES and FBIS) and 400,000 recent articles collected by FBIS in years 1999 and 2000.

Note queries in BBN9XLA and BBN9XLB contain both English terms and Chinese terms. To score a document against a query, two HMM scores were computed, one for the English query terms using the cross-lingual retrieval function, the other for the Chinese terms using the monolingual retrieval function. The two scores were multiplied to produce the final score for the document.

6 Official Retrieval Results

Table 2 shows the average precision for our submitted runs. What is striking is that all our cross-lingual runs have a higher score than our monolingual run. The results demonstrate that query expansion (BBN9XLA and BBN9XLB) improves retrieval performance, consistent with previous studies (Ballesteros and Croft, 1997).

BBN9MONO	BBN9XLA	BBN9XLB	BBN9XLC
0.2888	0.3401	0.3326	0.3099

Table 2: Retrieval results of submitted runs

7 Impact of query length and query expansion

Table 3 shows the impact of query expansion on cross-lingual retrieval performance. We show three versions of queries, short, medium and long. Short queries only use the words in the title field of the topics. Medium queries use title and description fields. Long queries use title, description and narrative. Query expansion improves performance for all query lengths. As expected, query expansion is more useful for short queries, and less useful for long queries. Three things are worth mentioning about the results. First, query expansion seems to neutralize the effect of query length. Without query expansion, the difference between short and long queries is 0.0669. After query expansion, it is reduced to 0.017. Second, English query expansion adds more than Chinese; apparently the benefit of a far larger corpus outweighs translation ambiguity. Third, while English expansion and Chinese expansion both improve retrieval performance, their combination does not improve performance further, except on the short queries. In fact, it is worse than either English expansion or Chinese Expansion alone for the medium queries. However, a query by query analysis shows that the surprising result is due to a statistical outlier in the retrieval results. The retrieval performance for topic 62 is 1.000 using English expansion and 0.3333 using both English and Chinese expansion. That query alone causes a difference of 0.0267 in average retrieval performance. Furthermore, topic 62 has only one relevant document; A small perturbation in the ranked output can cause a big change in retrieval performance. Under these circumstances, we cannot rule out the retrieval advantage of using both English and Chinese query expansion.

	No Expansion	Only English expansion terms	Only Chinese expansion terms	Both English & Chinese expansion terms
Short	0.2430	0.2991	0.2871	0.3231
Medium	0.2869	0.3282	0.3183	0.3038
Long	0.3099	0.3420	0.3326	0.3401

Table 3: Impact of query expansion on crosslingual retrieval performance

Table 4 shows the impact of query expansion on monolingual retrieval performance. As in cross-lingual retrieval, query expansion improves retrieval performance, but the amount of improvement is smaller.

	No Expansion	Expansion
Short	0.2299	0.2469
Medium	0.2476	0.2668
Long	0.2618	0.2888

Table 4: Impact of query expansion on monolingual performance

8 Impact of lexical sources on retrieval performance

The lexicon we used in our official runs is a combination of 4 lexical sources. Table 5 shows the contribution of each lexical source independently by reporting average precision without query expansion. The results show that the lexicon derived from the parallel corpus HKNews is the single most useful lexical resource; second is CETA, then LDC and last HKLaws. Each of these sources alone is significantly worse than the combined lexicon.

The experiment shows that different lexical sources can complement each other nicely. For our HMM-based approach, the results also show that the issue of lexicon completeness overrides that of translation ambiguity. On average, the combined lexicon has more than 1,000 Chinese translations per English query term. Even though this large figure is partly due to a few outliers, it does indicate there is a lot of translation ambiguity. The results indicate this does not have a big negative effect on retrieval performance.

	LDC only	CETA only	HKNews only	HKLaws only	ALL
Short	0.1491	0.1517	0.1875	0.1386	0.2430
Medium	0.1839	0.1944	0.2285	0.1395	0.2869
Long	0.1725	0.2126	0.2418	0.1441	0.3099

Table 5: Impact of lexical sources on average precision of retrieval. These results are without query expansion.

Another way to determine the value of a lexical source is to measure how much it contributes to the combined lexicon by removing the source from the combined lexicon and showing the impact on retrieval performance. The remaining sources were given equal weight. Table 6 shows that the most useful source is HKNews and the least useful HKLaws. In fact, removing HKLaws from the lexicon improves retrieval performance slightly. We think the reason is the domain mismatch between HKLaws and the TREC-9 Chinese corpus of news articles.

8.1 Comparing with TREC5 and TREC6 Queries

Although the TREC-5/6 Chinese corpus and TREC-9 corpus are both in Chinese, the former is in standard Chinese (Mandarin) and the latter in Cantonese. There are many differences in vocabulary between the two. As a result, using a bilingual lexicon for one dialect is sub-optimal for retrieval on a corpus in the other dialect. This effect can be seen when we compare retrieval performance using TREC-5/6 queries with TREC-9, as in Table 7. The LDC and CETA lexicons are better lexical resources than HKNews for TREC-5/6 but the opposite is true for TREC-9, probably because of the difference between the vocabularies in Mandarin and Cantonese. Had we had a bilingual lexicon for Cantonese, better retrieval results on TREC-9 may have been possible.

	ALL but LDC	All but CETA	ALL but HKNews	ALL but HKlaws
Short	0.2400	0.2298	0.1967	0.2462
Medium	0.2816	0.2678	0.2252	0.2950
Long	0.2924	0.2802	0.2506	0.3100

Table 6: Impact of removing a lexical source on average precision of retrieval. These results are without query expansion.

	LDC only	CETA only	HKNews only	HKLaws only
TREC-5 & 6 Medium	0.2897	0.3400	0.2496	0.1684
TREC-9 Medium	0.1839	0.1944	0.2285	0.1395

Table 7: Comparing TREC-5/6 and TREC-9

9 Utilizing English words in Chinese documents

Some Chinese documents in the TREC-9 corpus contain both English words and Chinese words. The English words are very useful for retrieval, for two reasons. First, they provide additional information about the content of the documents. Second, they can be utilized directly without translation, which invariably introduces errors. Such words were used in our submitted cross-lingual runs in the hope of improving retrieval. If we turned off this feature, the retrieval performance for BBN9XLC would be 0.3077 instead of 0.3099. Even though the difference is very small, we still think it is a desirable feature that can make a difference in a retrieval environment where such documents are common.

10 Monolingual Retrieval using Bigrams and Unigrams

Our official cross-lingual results are significantly better than our monolingual results. This anomalous result can be partly explained by the use of word-based indexing. As we discussed earlier, a word-based index is geared toward maximizing cross-lingual performance. For monolingual retrieval in Chinese, previous studies (Kwok, 1997) suggested that the best strategy may be to use bigrams. For comparison, we indexed the TREC-9 corpus using bigrams of Chinese characters and unigrams. Assuming a Chinese document is a sequence of Chinese characters, at each character position, we treat the bigram (current and the next characters) as a token. In addition, we also treat each character as a token. The resulting document is a bag of bigrams and unigrams. Stop words were discarded in the process. In a similar way, we processed the Chinese queries. Table 8 shows the monolingual results using bigrams and unigrams, together with our submitted results. Using bigrams and unigrams results in a huge improvement in monolingual performance. The results are also better than cross-lingual performance.

Bigrams. No Expansion	Bigrams. Query Expansion	BBN9MONO	BBN8XLA	BBN9XLB	BBN9XLC
0.3362	0.3779	0.2888	0.3401	0.3326	0.3099

Table 8: Using bigrams for monolingual retrieval

11 Conclusions

Our work was based on a previously reported HMM for retrieval (Miller et al., 1999); we extended that model from monolingual to cross-lingual retrieval. Several conclusions are suggested by the experiment:

1. As expected, query expansion improved short queries more than long queries. For this set of queries, it is interesting that the query expansion reduced the gap in (cross-lingual) performance between short and long queries from 25% relative without expansion to only 5% relative.
2. Quite surprisingly, with word-based indexing, all our cross-lingual runs were better than monolingual; the best cross-lingual run was 118% of monolingual. If we had used bigram indexing for monolingual performance, the best cross-lingual (word-based indexing) would have been 90% of the best monolingual (bigram based indexing).
3. Not surprisingly, the best bilingual resource was the one closest in dialect (Cantonese) and genre (news) to the document collection, even though it was automatically derived from a parallel corpus and highly ambiguous.
4. For our probabilistic model, coverage of the bilingual lexicon seems far more important than the degree of ambiguity in the lexicon.
5. Query expansion in English proved more valuable than query expansion in Chinese, in spite of the added ambiguity, perhaps because the English corpus for unsupervised relevance feedback was so much larger in English than for Chinese.

References

D. Miller, T. Leek and R. Schwartz, 1999. "A Hidden Markov Model Information Retrieval System." *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 214-221, 1999.

J. Allan, J. Callan, W. B. Croft, L. Ballesteros, D. Byrd, R. Swan, J. Xu, 1998. INQUERY Does Battle With TREC-6. In *TREC6 Proceedings*.

K. L. Kwok, 1997. Comparing Representations in Chinese Information Retrieval. *Proceedings of the 20th ACM SIGIR International Conference on Research and Development in Information Retrieval.*

L. Ballesteros and W.B. Croft, 1997. Phrasal translation and query expansion techniques for cross-language information retrieval. *Proceedings of the 20th ACM SIGIR International Conference on Research and Development in Information Retrieval*, pp. 84--91.

M. Porter, 1980. An algorithm for suffix stripping. In *Program*, 14(3), 130-137.

Acknowledgements

We wish to thank John Lafferty, whose WEAVER software was used to derive bilingual lexicons from parallel corpora. We also would like to thank Rich Schwartz for his advice.

The work reported here was supported in part by the Defense Advanced Research Projects Agency under contract number N66001-00-C8008. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the United States Government.

Appendix

Table 9 summarizes monolingual results in this report.

Name	Title	Desc	Narr	Query Exp	Words	Bigrams	Avg precision
BBN9MONO	x	x	x	x	x		0.2888
	x				x		0.2299
	x			x	x		0.2469
	x	x			x		0.2476
	x	x		x	x		0.2668
	x	x	x		x		0.2618
	x	x	x			x	0.3362
	x	x	x	x		x	0.3779

Table 9: Monolingual results. Words = Word-based index. Bigrams = index using bigrams and unigrams of Chinese characters.

Table 10 summarizes cross-lingual results in this report.

Name	Title	Desc	Narr	ChQE	EnQE	LDC	CETA	HK News	HK Laws	Avg Precision
BBN9XLA	x	x	x	x	x	x	x	x	x	0.3401
BBN9XLB	x	x	x	x		x	x	x	x	0.3326
BBN9XLC	x	x	x			x	x	x	x	0.3099
	x					x	x	x	x	0.2430
	x				x	x	x	x	x	0.2991
	x			x		x	x	x	x	0.2871
	x			x	x	x	x	x	x	0.3231
	x	x				x	x	x	x	0.2869
	x	x			x	x	x	x	x	0.3282
	x	x		x		x	x	x	x	0.3183
	x	x		x	x	x	x	x	x	0.3038
	x	x	x		x	x	x	x	x	0.3420
	x	x	x	x		x	x	x	x	0.3326
	x					x				0.1491
	x						x			0.1517
	x							x		0.1875
	x								x	0.1386
	x					x	x	x	x	0.2430
	x	x				x				0.1839
	x	x					x			0.1944
	x	x						x		0.2285
	x	x							x	0.1395
	x	x				x	x	x	x	0.2869
	x	x	x			x				0.1725
	x	x	x				x			0.2126
	x	x	x					x		0.2418
	x	x	x						x	0.1441
	x						x	x	x	0.2400
	x					x		x	x	0.2298
	x					x	x		x	0.1967
	x					x	x	x		0.2462
	x	x					x	x	x	0.2816
	x	x				x		x	x	0.2678
	x	x				x	x		x	0.2252
	x	x				x	x	x		0.2950
	x	x	x				x	x	x	0.2924
	x	x	x			x		x	x	0.2802
	x	x	x			x	x		x	0.2506
	x	x	x			x	x	x		0.3100

Table 10: Crosslingual results. Title=the title field, Desc=the description field, Narr=the narrative field, ChQE=Chinese expansion terms, EnQE=English expansion terms, LDC= the LDC lexicon, CETA= the CETA lexicon, HKNews=lexicon extracted from HKNews, HKLaws=lexicon extracted from HKLaws. A "x" indicates a topic filed, a lexical resource, or a query expansion type is used.

QALC - the Question-Answering system of LIMSI-CNRS

Olivier Ferret* Brigitte Grau* Martine Hurault-Plantet* Gabriel Illouz*
Christian Jacquemin* Nicolas Masson **
Paule Lecuyer **
*LIMSI-CNRS BP 133, 91403 ORSAY Cedex, FRANCE
**Bertin Technologies Parc d'Activités du Pas du Lac, 78180 Montigny-le-Bretonneux
{ferret,grau,gabrieli,jacquemin}@limsi.fr
{masson, lecuyer}@bertin.fr

1 Introduction

In this report we describe the *QALC system* (the Question-Answering program of the Language and Cognition group at LIMSI-CNRS) which has been involved in the QA-track evaluation at TREC9. The purpose of the Question-Answering track is to find the answers to a set of about 700 questions. The answers are text sequences extracted from the 5 volumes of the TREC collection. All the questions have at least one answer in the collection.

The basic architecture of *QALC* is composed of seven modules, two for the questions and four for the corpora, and a last pairing module which produces the sentences ranked by decreasing order of relevance (see Figure 1).

The *QALC* system relies mainly on genuine Natural Language Processing components. Most of the components take as input a tagged version of the documents. We use the *TreeTagger* for this purpose (Stein and Schmid, 1995). The system is based on the following modules:

Natural language question analysis The analysis of the questions relies on a shallow parser which spots discriminant patterns and assigns categories to the questions. The categories correspond to the types of entities which are likely to constitute the answer to this question.

Term extraction The term extractor is based on syntactic patterns which describe compound nouns. The maximal extension of these compounds is produced along with the plausible subphrases.

Search engine We tested two kinds of search engines: the search engine from ATT, by using only its outputs, and Indexal, a search engine from Bertin Technologies.

Automatic indexing & variant conflation Automatic indexing relies on *FASTR* (Jacquemin, 1999), a shallow transformational natural language analyzer which recognizes the occurrences and the variants of the terms produced by the term extraction module. Each occurrence or variant constitutes an index to the document which is ultimately used in the process of document ranking and in the process of question/document pairing.

Named entity recognition Similarly, named entities are recognized in the documents in order to build indexes which are used for measuring the degree of similarity between the questions and the document sentences. Named entities are extracted through lexico-syntactic patterns combined with significantly large lexical data.

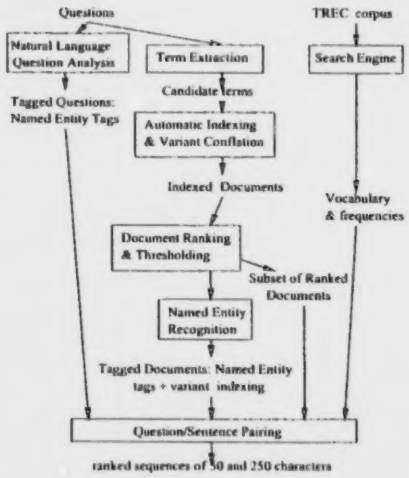

Questions
TREC corpus
Natural Language Question Analysis
Term Extraction
Search Engine
Tagged Questions: Named Entity Tags
Candidate terms
Automatic Indexing & Variant Conflation
Indexed Documents
Vocabulary & frequencies
Document Ranking & Thresholding
Subset of Ranked Documents
Named Entity Recognition
Tagged Documents: Named Entity tags + variant indexing
Question/Sentence Pairing
ranked sequences of 50 and 250 characters

Figure 1: Flowchart of QALC.

Document ranking & thresholding Documents are ranked according to a weighted measure of the indexes produced by the automatic indexing and variant conflation module. Only the *n* best ranked documents are selected. A further selection of the documents is made if a plateau can be recognized in the relevance curve of the documents.

Question/sentence pairing Finally, all the data extracted from the questions and the documents by the preceding modules is used by a pairing module to evaluate the degree of similarity between a document sentence and a question. The answers are extracted from the sentences that are chosen from the documents selected by the preceding module.

Our system is very similar to the system built for TREC8 (Ferret et al., 1999). Hence, we will only focus on the differences between these two systems in the following sections.

2 Natural Language Question Analysis

Question analysis is performed in order to assign features to questions and use these features for the pairing measurement between a query (question) and potential answer sentences (answer). Basically, question analysis allows the prediction of the kind(s) of answer, called *target* (for instance, ORGANIZATION). The retrieved documents (see Section 6) are labeled with the same tagset as the questions. During the pairing measurement, the more the question and a sentence share the same tags, the more they are considered as involved in a question-answer relation. For example:

Question: *How many people live in the Falklands?* → target = NUMBER

Answer: ... *Falklands population of* `<b_numex_TYPE=NUMBER> 2,100 <e_numex>` *is concentrated.*

The question analysis is based on a set of rules that uses syntactic and semantic criteria. The targets used are PERSON, ORGANIZATION, LOCATION (either CITY or PLACE), TIME-EXPRESSION (either DATE, TIME, AGE or PERIOD), and NUMBER (either LENGTH, VOLUME, DISTANCE, WEIGHT, PHYSICS or FINANCIAL). The target tags are hierarchised in order to offer more flexibility when searching for the answer. We have established 17 semantic classes, hierarchically structured as shown in Figure 2.

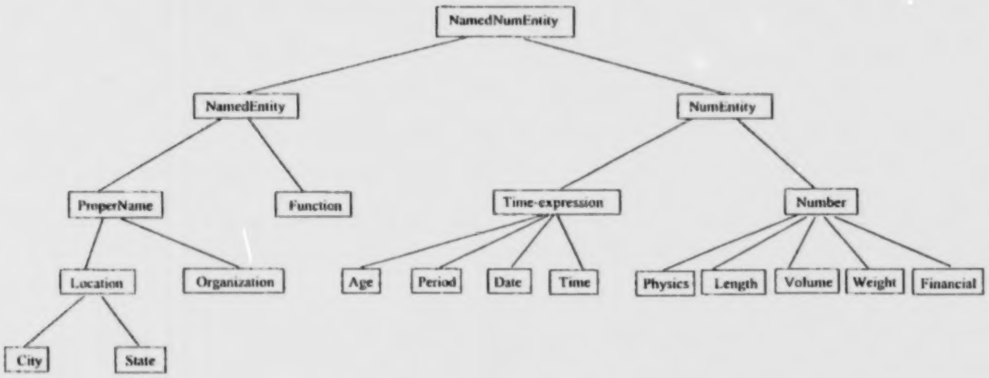


Figure 2: The hierarchy of target

3 Term Extraction

Terms are extracted from the questions and sentences through part-of-speech filtering. These terms are used by FASTR to index documents retrieved by the search engines.

The questions are first tagged with the help of the *TreeTagger*. Then, patterns of syntactic categories are used to extract terms from the tagged corpora. They are very close to those in (Justeson and Katz, 1995), but we do not include post-posed prepositional phrases. The pattern used for extracting terms is[1]:

$$(((((\mathrm{JJ}|\mathrm{N_N}|\mathrm{N_P}|\mathrm{V_{BG}})\ )?(\mathrm{JJ}|\mathrm{N_N}|\mathrm{N_P}|\mathrm{V_{BG}})\ (\mathrm{N_P}|\mathrm{N_N})))|(\mathrm{V_{BD}})|(\mathrm{N_N})|(\mathrm{N_P})|(\mathrm{CD})) \qquad (1)$$

The longest string is acquired first and substrings can only be acquired if they do not begin at the same word as the superstring. For instance, from the sequence ***name***$_{\mathrm{N_N}}$ ***of***$_{\mathrm{IN}}$ ***the***$_{\mathrm{DT}}$ ***US***$_{\mathrm{N_P}}$ ***helicopter***$_{\mathrm{N_N}}$ ***pilot***$_{\mathrm{N_N}}$ ***shot***$_{\mathrm{V_{BD}}}$ ***down***$_{\mathrm{RP}}$, the following four terms are acquired: ***US helicopter pilot***, ***helicopter pilot***, ***pilot***, and ***shoot***. We also keep all the lemmas corresponding to the single words of the question : ***US*** and ***helicopter***.

The mode of acquisition chosen for terms amounts to considering only the substructures that correspond to an attachment of modifiers to the leftmost constituents (the closest one). For instance, the decomposition of ***US helicopter pilot*** into ***helicopter pilot*** and ***pilot*** is equivalent to extracting the subconstituents of the structure [***US*** [***helicopter*** [***pilot***]]].

4 Search Engine

We tested the QALC system with the 200 questions that were proposed at the TREC8 Question-Answer task. One module of our system is the selection, through a search engine, of documents which may contain an answer to the question. We examined the results of three search engines, Zprise provided by NIST, ATT whose results for the TREC questions are provided by NIST, and Indexal placed at our disposal by Bertin Technologies, a French engineering company.

Zprise is a vectorial search engine developed by NIST. We used it with the following features : bm25idf weighting function of Okapi (Robertson E. and Beaulieu, 1999), stemming with the Porter algorithm, and no relevance feedback. Indexal is a pseudo-boolean search engine developed by Bertin Technologies. It is mostly used for information retrieval in smaller data bases. Indexal enlarges the request by use of stemming. A side effect of the retrieval method seems to be the preference given to large documents. Another feature is the use of the notion of affinity between words. The request then consists of a set of words that have to be found in a window of text of a given length.

---

[1] $\mathrm{N_N}$ are common nouns, $\mathrm{N_P}$ proper nouns and CD numeral determiners.

One goal of our tests was to determine the optimal number of documents to select from the retrieved documents. Indeed, having too many documents leads to a question processing time that is too long, but conversely, having too few documents reduces the possibility of obtaining the correct answer. Obviously, the other goal was to determine the best search engine, that is to say the one which gives the highest number of documents containing the answers.

4.1 Document Selection Threshold

We carried out four different tests with the Zprise search engine, respectively with 50, 100, 200, and 500 selected documents. At the end of TREC8, the NIST provided, for each question, a list of documents which contained the answer. We based our comparisons on this list of relevant documents . Table 1 shows the test results.

Number of selected documents	50	100	200	500
Number of questions for which all the relevant documents were retrieved	75	86	111	128
Number of questions for which some of the relevant documents were retrieved	116	98	82	66
Number of questions for which no relevant document was retrieved	19	16	7	6
Total number of relevant documentsthat were retrieved (total number: 1197)	702	820	931	1018

Table 1: Comparison between the numbers of relevant retrieved documents for different thresholds of selected documents

According to Table 1, the number of questions for which all the relevant documents were retrieved increases with the number of selected documents, but increases less between 200 et 500 selected documents. In the same way, the number of questions for which no document was retrieved is almost equal for 200 and 500 selected documents. Furthermore, the total number of relevant documents that were retrieved is less increasing between 200 and 500 selected documents (7% more) than between 50 and 100 selected documents (10% more). Generally speaking, the improvement of the search engine results tends to decrease beyond the threshold of 200 selected documents. For the TREC9 questions processing, we then choose the threshold of 200 documents selected which seemed to offer the best arrangement between the number of documents in which the answer may be found and the question processing time.

4.2 Search Engine Evaluation

We compared the results given by the three search engines for the 200 TREC8 questions and for a threshold of 200 selected documents. Table 2 gives the tests results.

Search engine	Indexal	Zprise	ATT
Number of questions for which all the relevant documents were retrieved	109	111	142
Number of questions for which some of the relevant documents were retrieved	73	82	52
Number of questions for which no relevant document was retrieved	18	7	6
Total number of relevant documentsthat were retrieved (total number : 1197)	814	931	1021

Table 2 : Compared performances of the Indexal, Zprise and ATT search engines

The ATT search engine revealed itself as the most efficient one according to the following three criteria: the largest number of questions for which all the relevant documents were retrieved, the lowest number of questions for which no relevant document was retrieved, and the most of relevant documents retrieved for all the 200 questions.

BEST COPY AVAILABLE

The evaluation results of our whole processing chain, tested with the TREC8 questions, were respectively 0.409 with Zprise, 0.452 with Indexal, 0.463 with ATT.

These different observations led us to choose two different search engines for the TREC9 QA task. The first one is ATT, for the obvious reason that it gives us the best results. The other is Indexal because, on the one hand, we can use it more freely as we have the software at our disposal, and, on the other hand, the Bertin Technology is in the process of improving its system.

5 Automatic Indexing and Document Ranking

The selection of relevant documents among the results given by the search engine relies on an NLP-based indexing composed of both single-word and phrase indexes and linguistic links between the occurrences and the original terms. The original terms are those extracted from the questions. The occurrences or variants of these terms are extracted by *FASTR* (Jacquemin, 1999), a transformational shallow parser for the recognition of term occurrences and variants. The detection of variants in the documents is based on rules, enabling morphological and semantic transformations.

The ranking of the documents relies on a weighted combination of the terms and variants extracted from the documents. The weighting scheme relies on a measure of quality of the different families of variations described in (Ferret et al., 1999): non-variant occurrences are better than morphological and morpho-syntactic variants, and semantic and morpho-syntactico-semantic variants receive the lowest weight. We also emphasize terms with proper names, since they are more reliable indexes than common names, and long terms are preferred over single ones, according to their number of words. Thus documents containing multiple word terms, either variant of initial terms or not, are better ranked than documents that contain scattered single words. We retain a maximum of 100 documents. However, when the weight curve presents a sudden slope, we only select documents before the fall, with a minimum set to 20.

6 Named Entity Recognition

Named entities are recognized in the documents in order to build indexes which are used for measuring the degree of similarity between the questions and the document sentences. Named entities receive one of the following types: `PERSON`, `ORGANIZATION`, `LOCATION (CITY or STATE)`, `NUMBER` (a time expression or a number expression, see Figure 2). They are defined in a similar way to the MUC task (Grishman and Sundheim, 1995) and recognized through a combination of

- lexical lookup (for syntactic or semantic tags on the single words) and rules which use these tags together with lexical elements; and
- dictionary lookup (the direct access to lists of named entities).

The three lists used for lexical lookup are CELEX (CELEX, 1998), a lexicon of 160,595 inflected words with associated lemma and syntactic category, a list of 8,070 first names (6,763 of which are from the CLR (1998) archive at New Mexico State University) and a list of 211,587 family names from the CLR archive at New Mexico State University. The recognition of location expressions was improved by integrating the work of (Illouz, Jacquemin, and Habert, 1999).

7 Question/Sentence Pairing

This section first presents the module that selects for each question a list of five ranked sentences in which the *Answer Extraction* module then tries to find the five required answers. This module relies on the results of all the preceding modules:

- each question is assigned a set of terms and one or several categories according to its focus;

- a set of documents is selected for each question. In each of them, named entities and terms extracted from the questions are tagged.

Although they share a large number of features, the *Question/Sentence Pairing* module for the 250 character task is not the same as the one for the 50 character task. The first one is identical to the *Pairing* module used in the TREC8 *Qalc* system while the second one is a new one that aims at being more precise. Nevertheless, these two modules are based on the same principle: we compare each sentence from the selected documents for a question to this question and we always keep the five sentences that are the most similar to the question.

7.1 The TREC8 *Pairing* Module

In the TREC8 *Pairing* module, questions and document sentences are transformed into vectors. Then these vectors are compared by computing a similarity measure. Such vectors contain the most significant words of the primary sentences or questions, i.e. mainly their content words (as nouns, verbs and adjectives). Each word in a vector is weighted according to its importance in relation to the *Question/Answering* corpus by using the *tf.idf* weighting policy, as it is often done in Information Retrieval. These vectors also contain two kinds of linguistic features: terms and named entities. These linguistic features are considered in the vectors as if they were significant words.

Terms Each term that has been extracted by the term extractor described in Section 3 has an unique identifier that is used for marking the occurrences and the variants of this term both in the questions and in the document sentences. In the sentences, this identifier is associated with a score that reflects the distance between the found variant and its reference form in the question vector (see Section 5). In the question vector, we add the term identifier with a default weight.

Named entities Each recognized named entity is marked with a specific tag according to its type (see Section 6). On the other hand, the kind of the answer expected for each question is determined by the *Question Analysis* module. Thus, for a question, we add the tag(s) of the expected type(s) of the answer to its vector and for a sentence, we add the tags of the named entities that have been recognized in the sentence to its vector. In both cases, each tag is given a fixed weight.

The comparison between a sentence vector V_d and a question vector V_q, both enriched with linguistic features, is then achieved by computing the following similarity measure:

$$sim(V_q, V_d) = \frac{\sum_i wd_i}{\sum_j wq_j} \tag{2}$$

with wq_j, the weight of a word in the question vector and wd_i, the weight of a word in a sentence vector that is also in the question vector. This measure evaluates the proportion and the importance of the elements (words or linguistic features) in the question vector that are present in the sentence vector with regards to all the words in the question vector.

We also take into account the difference in length between a question and a document sentence. This criterion is used as a secondary key for sorting the sentences that are selected as possible answers to a question: if two sentences have the same similarity value, the shortest sentence is ranked first.

7.2 The TREC9 *Pairing* Module

The *Pairing* module used this year for the 50 character task differs from the above one on two main points. First, it makes no distinction between words and terms. It only deals with terms and considers words as mono-terms. Second, it does not compute the similarity between a question and a sentence globally by gathering all their features in a vector. On the contrary, question/sentence similarity is evaluated for each kind of features and the results of this evaluation are then aggregated in relation to the importance of the type of the features. Three kinds of features are taken into account:

- the presence in the sentence of the terms extracted from the question;
- the presence in the sentence of named entities whose type fits the expected type of the answer;
- the length of the shortest part of the sentence that contain all the terms that are terms of the question.

The first change is intended to simplify the overall process while the second one aims at increasing the tuning capabilities of the pairing module.

7.2.1 Term Similarity

The term similarity of a sentence in relation to a question is given by the sum of the weights of the terms extracted from the question that are present in the sentence, either as they are or as variants. The weight of such a term T takes into account three factors:

- the score given by *FASTR* to the term T. This score depends on what kind of variant of T was recognized (see Section 5);
- the fact that T is or not a proper noun. We actually consider that proper nouns are more significant than the terms built with common words;
- the specificity of the term T. As Kozima (1993), we evaluate this specificity by using the significativity of T, which corresponds to its normalized information with regards to a corpus. In our case, the reference corpus is the *LA Times* part of the *Question Answering* corpus. As they are considered as very specific, multi-terms and proper nouns are given the maximum significativity value, which is equal to 1.

The weight of a term T combines these three factors into the following formula:

$$term\_weight(T) = fastr\_score(T) + (\frac{fastr\_score(T)}{2} \bullet np\_modulator(T) \bullet significativity(T)) \qquad (3)$$

1. $fastr\_score(T)$: score given by *Fastr* to the term T;
2. $np\_modulator(T)$: proper noun modulator. It is equal to 2 for proper nouns and equal to 1 for the other terms;
3. $significativity(T)$: significativity of the term T.

Roughly speaking, the weight of a term T is equal to its *Fastr* score plus a modulation of the half of that score in relation to the type and the specificity of T. Moreover, if a term of a question has several occurrences in a sentence, only the occurrence that has the greatest weight is kept.

7.2.2 Answer Length Score

The length score of a sentence aims at favoring sentences in which the terms of the question are grouped in a small area. Its presence is justified by the presence in the *Question Answering* corpus of a large proportion of newspaper articles, which often contain long sentences.

The length score is simply equal to the number of words in the smallest part of the sentence that gathers all the terms of the question that were recognized for this sentence.

7.2.3 Named Entity Similarity

The named entity similarity measure of a sentence in relation to a question takes into account two factors:

- the presence in the sentence of named entities that correspond to the expected type of the answer;
- the distance of these named entities to the terms of the question that were recognized in the sentence.

The evaluation of the first factor relies on the named entity hierarchy presented in Section 2. Questions are always tagged with the more specific types of this hierarchy. But our named entity tagger can not always be so accurate. Hence, we search in the document sentences not only the named entities having the type of the answer but also the named entities having a more general type. Of course, the score of a named entity decreases as its distance from the expected type, i.e. the number of levels that separate them in the hierarchy, increases. If several named entities in a document sentence are found to be compatible with the answer type, we only keep the one having the greatest score.

The second factor is justified by the length of the sentences in the *Question Answering* corpus. We suppose that a correct answer is more likely to be found if the named entity that fits the type of the answer is close to the recognized terms of the question. We take as reference for these terms the part of the sentence that is delimited for the computation of the answer length score. We consider that if a named entity occurs more than 4 words away from the beginning or the end of this part of sentence, it has few chance to be related to the question.

Finally, the named entity similarity measure is given by the following formula:

$$\begin{aligned} ne\_similarity &= ne\_hierarchy\_level\_number - distance(question\_type, best\_document\_ne) \\ &\quad + answer\_proximity(best\_document\_ne) \end{aligned} \tag{4}$$

1. *ne_hierarchy_level_number*: number of levels of the named entity hierarchy. In our case, it is equal to 3;
2. *distance(question_type, best_document_ne)*: number of levels between the answer type and the type of the best compatible named entity found in the document sentence;
3. *answer_proximity(best_document_ne)*: proximity between the best named entity and the terms of the question. In our case, this is a binary value: 1 if the best named entity fulfills the proximity conditions; 0 otherwise.

7.2.4 Global Similarity

As in the TREC8 *Pairing* module, the global similarity measure in this module is used to rank the document sentences that may contain the answer to a question. This measure consists of aggregating the three dimensions we have presented above according to the following principles.

First, we favor the term similarity: if a sentence *S1* has a term similarity that is significantly greater than the term similarity of a sentence *S2*, *S1* is ranked on top of *S2*. By ***significantly greater***, we mean more precisely that $term\_similarity(S1) > term\_similarity(S2) + similarity\_equivalence$. For the runs we submitted, *similarity_equivalence* was equal to 0.1. When the term similarity is too ambiguous, we rely on named entity similarity: if *S1* has greater named entity similarity than *S2*, *S1* is ranked on top of *S2*. As there are not many possible values for that kind of similarity, this criterion may also be ambiguous. In such a case, we come back to the term similarity criterion, but with a smaller *similarity_equivalence* value: this one was equal to 0.05 for our evaluation runs. Finally, if there is still an uncertainty, we use the answer length score: we choose the sentence that has the lowest length score.

7.3 Answer Extraction

The selection of a subpart of a sentence longer than 250 characters is simply done by reducing it by its two ends. The extraction of a short answer depends on the presence or not of a tag that may correspond to the kind of answer, if it is known. We retain either the exact tag proposed by the question analyzer or a more general one in our hierarchy. If such a tag is found in the sentence itself, or just besides in one of the two contiguous sentences, we select the tagged expression. When there is no tag in the sentence or no requested tag found by the question analyzer, we select the longest part of the sentence that does not contain any term.

8 Results and Analysis

We sent to TREC9 three different runs. Two of those runs give answers of 250 characters and use the same processing chain except for the search engine module which uses for one run ATT, and for the other run, Indexal. The third run gives answers of 50 characters length and is the result of a different ending module. The search engine used for the third run is ATT. Results are consistent with our previous analysis. Indeed, the run with ATT search engine gives slightly better results (0,407 strict et 0,414 lenient) than those obtained with the Indexal search engine (0,375 strict et 0,382 lenient). Table 3 sums up the number of answers found by our two runs.

Rank of the retrieved answer	1	2	3	4	5	No answer retrieved
Run using ATT (strict)	216	73	48	23	15	307
Run using Indexal (strict)	187	78	50	35	22	310
Run using ATT (lenient)	221	72	50	23	16	300
Run using Indexal (lenient)	190	81	50	34	23	304

Table 3 : Number of retrieved answers, by rank, for the two runs at 250 caracters

The results given in Table 3 lead us to the following remarks : the run using the ATT search engine gives more answers at rank 1 than the run using Indexal. Conversely, this last run gives more answers in lower ranks. Therefore, the score difference between these two runs seems to result more from a better ranking of the correct answers than from the slightly different number of retrieved answers.

We then analysed the overlap between the answers of the two runs. These runs give rather different results, as they have only partial overlapping : for 246 same questions, none of them gives the correct answer at any rank (among about 310 not found for each of them). The other 60 questions for which they do not have the correct answer are different for each run. Furthermore, only 169 answers are found at the same rank for the two runs, among about 370 retrieved answers, e.g. a little less than a half. As shown in Table 3, answers given by the run using ATT have a better rank than those given by the run using Indexal: 162 questions are ranked better through ATT while only 105 question are ranked better by Indexal.

9 Conclusion and Future Developments

Our participation to the *Question Answering* track this year was guided by two purposes, mainly concerned with the support to the QA task and not with the QA task itself:

- transforming our previous set of modules into an actual QA system that can be easily used in order to test different ideas;
- integrating in our QA system a search engine that we can tune.

We also improved the *Question/Answer Pairing* and the *Answer Extraction* modules but these ones still have to be perfected as it is proved by our results for the 50 character task.

Among the future developments that we are considering for our next participation in the QA-track are:

- answer unit could be enlarged and position of indexes inside a document could be accounted for in order to focus on the units that gather the largest number of indexes and which are more likely to provide the answer;

- term acquisition could be improved through a disambiguation of long noun phrases and a better part-of-speech tagging of the questions;
- named entity recognition could be improved through machine learning techniques (Baluja, Mittal, and Sukthankar, 1999);
- we achieved some tests about using topic resources for query expansion, with some promizing results that we could not exploit this year;
- although we focused this year on architecture issues, there is still work to do in this area, especially about having coherent linguistic annotations among all our tools, such as in the ATLAS framework (Bird et al., 2000), for example.

References

Baluja, Shumeet, Vibhu O. Mittal, and Rahul Sukthankar. 1999. Applying machine learning for high performance named-entity extraction. In *Proceedings, PACLING'99*, pages 365–378, Waterloo, CA.

Bird, Steven, David Day, John Garofolo, John Henderson, Christophe Laprun, and Mark Liberman. 2000. Atlas: A flexible and extensible architecture for linguistic annotation. In *Second International Conference on Language Resources and Evaluation*, pages 1699–1706.

CELEX. 1998. *www.ldc.upenn.edu/ readme_files/celex.readme.html*. Consortium for Lexical Resources, UPenn.

CLR. 1998. *http://crl.nmsu.edu/cgi-bin/Tools/CLR/clrcat#D3*. Consortium for Lexical Resources, NMSU, New Mexico.

Ferret, O., B. Grau, G. Illouz, C. Jacquemin, and N. Masson. 1999. QALC - the Question-Answering program of the Language and Cognition group at LIMSI-CNRS. In *TEXT RETRIEVAL CONFERENCE (TREC-8)*, pages 387–404, Columbia, MD. NIST Special publication.

Grishman, R. and B. Sundheim. 1995. Design of the muc-6 evaluation. In NIST, editor, *the Sixth Message Understanding Conference (MUC-6)*, Columbia, MD. NIST, Morgan-Kauffmann Publisher.

Illouz, G., C. Jacquemin, and B. Habert. 1999. Repérage des entités nommées (REN) dans des transcriptions de documents parole. Technical Report 99-18, LIMSI-CNRS.

Jacquemin, Christian. 1999. Syntagmatic and paradigmatic representations of term variation. In *Proceedings, ACL'99*, pages 341–348, University of Maryland.

Justeson, John S. and Slava M. Katz. 1995. Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, 1(1):9–27.

Kozima, H. 1993. Text segmentation based on similarity between words. In *31th Annual Meeting of the Association for Computational Linguistics (Student Session)*, pages 286–288.

Robertson E., Walker S. and M. Beaulieu. 1999. Okapi at trec7: automatic ad hoc, filtering, vlc and interactive. In *Seventh Text Retrieval Conference (TREC-7)*.

Stein, Achim and Helmut Schmid. 1995. Etiquetage morphologique de textes français avec un arbre de décision. *t.a.l.*, 36(1-2):23–36.

SPOKEN DOCUMENT RETRIEVAL FOR TREC-9 AT CAMBRIDGE UNIVERSITY

S.E. Johnson† *, *P. Jourlin*‡ **, *K. Spärck Jones*‡ & *P.C. Woodland*†

†Cambridge University Engineering Department,Trumpington Street, Cambridge, CB2 1PZ, UK.
Email: `{sej28,pcw}@eng.cam.ac.uk`
‡Cambridge University Computer Laboratory, Pembroke Street, Cambridge, CB2 3QG, UK.
Email: `{pj207,ksj}@cl.cam.ac.uk`

ABSTRACT

This paper presents work done at Cambridge University for the TREC-9 Spoken Document Retrieval (SDR) track. The CU-HTK transcriptions from TREC-8 with Word Error Rate (WER) of 20.5% were used in conjunction with stopping, Porter stemming, Okapi-style weighting and query expansion using a contemporaneous corpus of newswire. A windowing/recombination strategy was applied for the case where story boundaries were unknown (SU) obtaining a final result of 38.8% and 43.0% Average Precision for the TREC-9 short and terse queries respectively. The corresponding results for the story boundaries known runs (SK) were 49.5% and 51.9%. Document expansion was used in the SK runs and shown to also be beneficial for SU under certain circumstances. Non-lexical information was generated, which although not used within the evaluation, should prove useful to enrich the transcriptions in real-world applications. Finally, cross recogniser experiments again showed there is little performance degradation as WER increases and thus SDR now needs new challenges such as integration with video data.

1. INTRODUCTION

With the ever-increasing amount of digital audio data being produced, it is becoming increasingly important to be able to access the information contained within this data efficiently. Spoken Document Retrieval (SDR) addresses this problem by requiring systems to automatically produce pointers to passages in a large audio database which are potentially relevant to text-based queries. The systems are formally evaluated within TREC using relevance assessments produced by humans who have listened to the audio between previously established manually-defined "story" boundaries. A transcription generated manually is also provided for a reference run to give an approximate upper-bound on expected performance.

The natural way to allow easy indexing and hence retrieval of audio information is to represent the audio in a text format which can subsequently be searched. One such method is to represent the speech present in the audio as a sequence of sub-word units such as phones; generate a phone sequence for the text-based query; and then perform fuzzy matching between the two. (see e.g. [4, 17]) The fuzzy phone-level matching allows flexibility in the presence of recognition errors and out of vocabulary (OOV) query words can potentially find matches. However, this approach still requires a method of generating phone sequences from the query words (usually a dictionary); it cannot easily use many standard text-based approaches, such as stopping and stemming; and performance on large scale broadcast news databases, such as those used within the TREC-SDR evaluations is generally poor[8].

With the recent improvements in the performance and speed of large vocabulary continuous speech recognition (LVCSR) systems, it is possible to produce reasonably accurate word based transcriptions of the speech within very large audio databases. This allows standard text-based approaches to be applied in retrieval, and means that a real user could easily browse the transcripts to get an idea of their topic and hence potential relevance without needing to listen to the audio. (see e.g. [27]). The inclusion of a language model in the recogniser greatly increases the quality of the transcriptions over the phone-based approach, and the overall performance of word-based systems has outperformed other approaches in all previous TREC-SDR evaluations [8]. OOV words do not currently seem to present a significant problem provided that suitable compensatory measures are employed [28] and rolling language models have been investigated (see e.g. [3]) as a way to adapt to changing vocabularies as the audio evolves.

Several methods to compensate for the errors in the automatically generated transcriptions have been devised. Most of these use a contemporaneous text-based news-wire corpus to try to add relevant non-erroneous words to the query (e.g. [1, 12]) or documents (e.g. [22, 23, 12]) although other approaches are also possible (e.g. the machine-translation approach in [5]). These methods have proven very successful even for high error rate transcriptions [16], so the focus of SDR has generally switched to trying to cope with continuous audio streams, in which no "document" boundaries are given[1]. This story-boundary-unknown (SU) task is the main focus of the TREC-9 SDR evaluation.

---

* Now Sue Tranter, Dept. of Engineering Science, Oxford, OX1 3PJ, UK : `sue.tranter@eng.ox.ac.uk`

** Now at Laboratoire d'Informatique de l'Universite d'Avignon : `pierre.jourlin@lia.univ-avignon.fr`

[1] Or at least where topic boundaries are not available within the global boundaries of a newscast.

Our overall approach involves generating a word-level transcription and dividing it into overlapping 30 second long windows. Standard stopping, stemming and Okapi-weighting are used during retrieval with query expansion from a contemporaneous newswire collection, before merging temporally close windows to reduce the number of duplicates retrieved.

This paper describes the Cambridge University SDR system used in the TREC-9 SDR evaluation. Sections 1.1 and 1.2 describe the tasks and data for the evaluation in more detail. The problem of extracting non-lexical information from the audio which may be helpful for retrieval and/or browsing is addressed in section 2 and the transcriptions used are described in section 3. Development for the SU runs is given in section 4, with results from the final system on all transcriptions and query sets given in section 5. The effects of using non-lexical information in retrieval are investigated in section 6 and a contrast for the case where story boundary information is known (SK) is given in section 7. Finally conclusions are offered in section 8.

1.1. Description of TREC-9 SDR Tasks

The TREC-9 SDR evaluation [6] consisted of two tasks. For the main story-boundary-unknown (SU) task, the system was given just the audio for each news episode (e.g. entire hour-long newscasts) and had to produce a ranked list of episode:time stamps for each text-based query. The scoring procedure involved mapping these stamps to manually defined story-IDs, with duplicate hits being scored as irrelevant, and then calculating Precision/Recall in the usual way[2].

The two differences from this task to the TREC-8 SDR SU evaluation task [8, 12] are firstly that for TREC-9, all the audio was judged for relevance (including e.g. commercials) and secondly that non-lexical information (such as the bandwidth/ gender /speaker-ID, or the presence of music etc.) that was automatically detected by the speech recognition system could be used in addition to the word-level output at retrieval time. A contrast run (SN) was required without the use of the non-lexical information, if it had been used within the SU run, to allow the effect of this additional information to be seen.

Another contrast run where manually-defined story boundaries were provided (SK) allowed the degradation from losing the story boundary information to be evaluated. This is the same as the primary task in the TREC-8 SDR evaluation. Sites had to run on their own transcriptions (`s1`), a baseline provided by NIST (`b1`) and the manually-generated reference (`r1`)[3].

1.2. Description of Data

The audio data for the document collection was the same as that used in the TREC-8 SDR evaluation, namely 502 hours (~4.5M words) from 902 episodes of American news broadcast between February and June 1998 inclusive. The SK runs took a subset of ~3.8M words divided into 21,754 manually defined "stories" to give an average document length of ~170 words.

The queries used for development (TREC-8) and evaluation (TREC-9) are described in Table 1. Two sets of queries were used, namely *short* (corresponding to a single sentence) and *terse* (approximately 3 key words). The query sets corresponded to the same original information needs and thus the same relevance judgements were used in both cases. The introduction of terse queries was new for TREC-9, and was intended to model the keyword-type query used in many WWW search engines. Since there were no existing terse development queries, *terse forms of the TREC-8 queries were developed in house and thus are not the same as those used by other sites.*

	Dev (TREC-8)	Eval (TREC-9)
Num. Queries	49	50
Ave. # Words in Query	13.6 (s) 2.4 (t)	11.7 (s) 3.3 (t)
Ave. # Distinct Terms per Q	6.6 (s) 2.3 (t)	5.6 (s) 2.9 (t)
Ave. # Rel Docs	37.1	44.3

Table 1: Properties of query and relevance sets.(s=short t=terse)

The contemporaneous parallel text corpus used for query and document expansion consisted of 54k newswire articles (36M words) from January to June 1998. Although significantly smaller than that used by some other sites (e.g. 183k articles in [24]), in previous work we found that increasing the parallel corpus size to approximately 110k articles did not help performance [16]. The corpus, summarised in Table 2, consisted of the (unique) New York Times (NYT) and 20% of the Associated Press (APW) articles from the TREC-8 SDR Newswire data enhanced with some LA Times/Washington Post (LATWP) stories and was evenly distributed over the whole time period.

Source	LATWP	NYT	APW	Total
Num. Stories	15923	20441	17785	54149
Ave. # Words in Doc.	685	885	385	662

Table 2: Description of the Parallel Corpus.

2. GENERATING NON-LEXICAL INFORMATION

Audio contains much more information than is captured simply by transcribing the words spoken. For example, the way things are said, or who said them can be critical in understanding dialogue, and many non-speech events (such as music, applause, sudden noises, silence etc.) may also help the listener follow what was recorded. Current speech recognisers can automatically recognise many of these things, such as the speaker ID or gender (e.g. [13]) and the presence of music, noise and silence etc. (e.g. [21]), but the speech-recognition-transcription (SRT) format used in the SDR evaluations does not support the inclusion of such additional information. For TREC-9 a new Segmentation Detection Table (SDT) file was allowed [6], which represented various audio phenomena found during recognition in a text-based format which could be used at retrieval time.

---

[2] Precision and Recall were calculated with respect to whole stories, rather than a more natural passage-based approach for logistic reasons.

[3] See section 3 for more details.

There are two main uses for such non-lexical information, namely to increase retrieval performance and to help navigation/browsing in real SDR applications. The TREC-9 SDR evaluation only allowed the former to be properly evaluated, but the latter is equally important in real world applications, and tags should not be thought to be irrelevant just because they were not used in the retrieval stage of the system [18].

Non-lexical information can be used to help SU retrieval in two main ways. Firstly some information about broadcast structure including potential locations of commercials and story boundaries can be postulated from audio cues such as directly-repeated audio sections, changes in bandwidth/speaker or the mean energy in the signal. Secondly properties such as the presence of music, background noise or narrowband speech can be used to identify portions of transcription which are potentially less reliable than normal.

Table 3 shows the tags generated, whilst the next section explains how these were produced and section 6 discusses their effect on retrieval performance.

Tag Number	(high)-Energy 19,882	Repeat 7,544	Commercial 5,194
Segment 142,914	Gender 57,972	Bandwidth 49,542	Nospeech 15,700

Table 3: Non-lexical tags generated for TREC-9.

2.1. Segment, Gender, Bandwidth and Nospeech

The first stage of our speech recognition system consists of an audio segmenter. Initially the data is classified into wideband speech, narrowband speech or pure music/noise, giving the `bandwidth` and `nospeech` tags respectively. The labelling process uses Gaussian mixture models and incorporates MLLR adaptation. A gender-dependent phone recogniser is then run on the data, and the smoothed `gender` change points and silence points are used in the final segmentation, hence generating the `segment` tags. More details can be found in [12] and [11].

2.2. Energy

Signal energy can help to indicate the presence of commercials. The average normalised log energy (NLE)[4] for the TREC-7 and January TDT-2 data, given in Table 4, shows that in general commercials have a higher mean energy content than news.

	TREC-7 data			January TDT-2 data	
Br.	Story	Filler	Comm.	News	Comm.
ABC	-2.82	-2.82	-1.95	-2.98	-2.22
CNN	-2.22	-2.21	-1.69	-2.27	-2.08
PRI	-2.40	-2.63	-1.84	-2.61	-2.48

Table 4: Average normalised log-energy for TREC-7 and January TDT-2 data for Stories, Fillers and Commercials.

By windowing the audio and comparing the NLE for each 5s window to a threshold, it is possible to generate a crude indicator of where commercials might be occurring. Imposing a minimum length restriction on the postulated commercials can be used to reduce the false alarm rate. Table 5 shows the results of applying such a system on the development (January TDT-2) and test (TREC-9) data. Whilst the method does pick out relatively more commercials than news stories, it is not accurate enough in itself to be used during retrieval, and would need to be combined with other cues for more reliable commercial identification. Tags were generated using a threshold of 10dB (NLE=-1.3), but these were not used in the retrieval system for the reason mentioned.

θ	ml	ABC	PRI	CNN
-1.5	-	36.9@3.2	37.4@15.5	59.2@13.9
-1.3	-	22.0@1.5	27.6@ 9.5	44.9@ 7.0
-1.3	20s	9.5@0.2	15.6@ 4.1	23.0@ 1.3

a) Development data (January TDT-2)

θ	ml	ABC	PRI	CNN	VOA
-1.5	-	39.3@3.4	49.2@26.1	53.0@13.7	18.8@4.8
-1.3	-	23.7@1.7	40.0@17.6	41.5@ 7.2	13.9@2.7
-1.3	20s	8.6@0.2	25.0@ 7.6	21.5@ 1.5	3.7@1.0

b) TREC-9 test data

Table 5: Percentage non-story @ story rejection when using a threshold, θ, on the normalised log energy for 5s windows, including restricting the minimum length, ml.

2.3. Repeat and Commercial

Direct audio repeats (i.e. re-broadcasts) were found using the technique described in [14], by comparing all the audio (across the entire 5 months) from each broadcaster. Commercials were postulated in a similar way to that described in [12], by assuming that segments which had been repeated several times were commercials and that no news portion of less than some smoothing length could exist between them. Table 6 shows the results from applying the parameter set used in the evaluation (C-E) and a less conservative run (C-2) as a contrast. The numbers for our TREC-8 commercial detection system are given for comparison.

	Time (h)		TREC-8	TREC-9	
Br.	N-St	St.	C-E	C-E	C-2
ABC	19.5	42.9	65.5@0.02	79.8@0.01	83.3@0.13
CNN	73.3	170	35.7@0.46	62.4@0.43	69.8@0.62
PRI	11.6	81.5	16.6@0.10	24.5@0.14	28.0@0.19
VOA	9.4	92.9	5.0@0.04	7.2@0.09	8.1@0.11
ALL	114	388	36.3@0.23	57.0@0.24	62.7@0.35

Table 6: Overall time and percentage of non-stories @ stories rejected using both the TREC-8 and TREC-9 commercial detection systems with a less conservative C-2 run for comparison.

Detection performance with this strategy is very impressive, with over half the adverts being identified for negligible loss of news content. Removing these postulated commercials automatically before retrieval was earlier shown not only to reduce the amount of processing necessary but also to significantly improve performance on the TREC-8 data [15]. The improvement from the

---

[4] NLE is related to the dB from the maximum energy in the episode by: `ln10 • dB = 10 • ( 1 - NLE )`

TREC-8 to the TREC-9 commercial detection system is due to the change in rules which allows both segments for any given match to be noted within the SDT file[5].

3. TRANSCRIPTIONS

3.1. `s1` Transcriptions

The transcriptions used for our `s1` runs were those we generated for the 1999 TREC-8 SDR evaluation. A summary of the system is shown in Figure 1 and a detailed description can be found in [12]. The system ran in 13xRT[6] and gave a Word Error Rate (WER) of 15.7% on the November 1998 Hub4 eval data and 20.5% on the 10-hour scoring subset of the TREC-8 data.

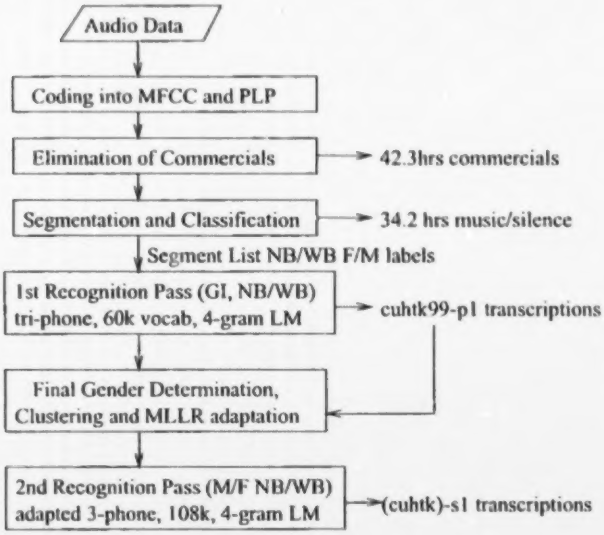

Audio Data
Coding into MFCC and PLP
Elimination of Commercials
42.3hrs commercials
Segmentation and Classification
34.2 hrs music/silence
Segment List NB/WB F/M labels
1st Recognition Pass (GI, NB/WB) tri-phone, 60k vocab, 4-gram LM
cuhtk99-p1 transcriptions
Final Gender Determination, Clustering and MLLR adaptation
2nd Recognition Pass (M/F NB/WB) adapted 3-phone, 108k, 4-gram LM
(cuhtk)-s1 transcriptions

Figure 1: System used to generate transcriptions.

3.2. Other Available Transcriptions

Manually generated closed-caption transcriptions[7] were available for the stories within the SK part of the evaluation from TREC-8 [8]. Word-level time stamps for these portions were produced by LIMSI using forced alignment after some text normalisation. Reference transcriptions were also made for the remaining untranscribed portions of the data by NIST using ROVER on the available TREC-8 ASR transcriptions [7]. The subsequent reference `r1` was thus considerably different to the corresponding set of reference transcriptions for TREC-8.

Additional transcriptions were made available for the TREC-9 SDR runs. The baseline cases from TREC-8 SDR produced by NIST using the BBN Rough'N'Ready recogniser [3] were re-released with `b1` from TREC-8 becoming `cr-nist99b1`, whilst `b2` from TREC-8 became the baseline `b1` for TREC-9. The TREC-8 transcriptions from Sheffield [2] and LIMSI [9] were re-released as `cr-shef-s1` and `cr-limsi-s1`, whilst both sites provided new (higher quality) transcriptions named `cr-shef-s2` [2] and `cr-limsi-s2` [10] respectively. The WER for these sets of transcriptions on the 10hr TREC-8 scoring subset of the corpus are shown in Table 7.

Recogniser	Corr.	Sub.	Del.	Ins.	WER
r1	91.9	2.5	5.6	2.2	10.3
(cuhtk-)s1	82.4	14.0	3.7	2.9	**20.5**
cr-limsi-s2	82.1	14.2	3.7	3.3	21.2
cr-limsi-s1	82.0	14.6	3.4	3.5	21.5
cr-cuhtk99-p1	77.3	18.5	4.2	3.9	26.6
b1	76.5	17.2	6.2	3.2	26.7
cr-nist99b1	75.8	17.8	6.4	3.3	27.5
cr-shef-s2	74.6	20.0	5.4	3.8	29.2
cr-shef-s1	71.9	22.0	6.1	3.9	32.0

Table 7: WER on TREC-8 10 hour scoring subset of eval. data.

4. SU DEVELOPMENT

4.1. The Basic System

The basic framework for the SU system, shown in Figure 2, is similar to our TREC-8 system [12]; but it does not enforce boundaries at proposed commercial breaks, it uses a different method of performing query expansion and is simpler in not having part-of-speech query weighting, semantic poset indexing or parallel collection frequency weighting.

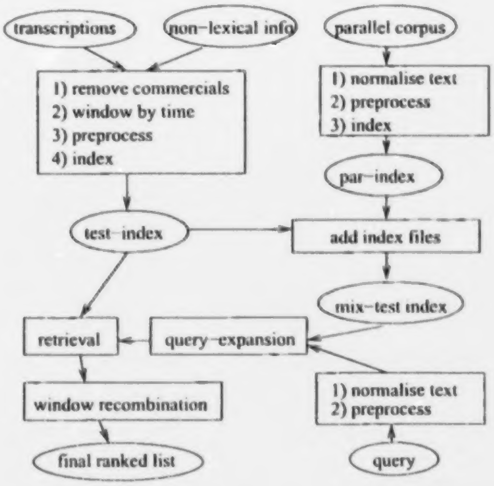

transcriptions
non-lexical info
parallel corpus
1) remove commercials
2) window by time
3) preprocess
4) index
1) normalise text
2) preprocess
3) index
par-index
test-index
add index files
mix-test index
retrieval
query-expansion
window recombination
1) normalise text
2) preprocess
final ranked list
query

Figure 2: Framework for the SU system.

The transcriptions were first filtered, removing all words which occurred within periods labelled as `commercial` in the non-lexical file (see section 2.3). Windows of 30s length with an inter-window shift of 15s were then generated to divide up the continuous stream of transcriptions.

---

[5] In TREC-8, the commercial detection was done pre-recognition in an *on-line* manner i.e. you could not add information about past events retrospectively.

[6] On a Pentium III 550MHz processor running Linux.

[7] Closed-caption transcriptions often use paraphrases or summaries hence giving a significant WER.

Text-normalisation was applied to the query and parallel corpus to minimise the mismatch between the ASR transcriptions and the text-based sources. Preprocessing including mapping phrases and some stemming exceptions, punctuation removal, stop word removal and stemming using Porter's algorithm, for all documents and queries. The stoplist included numbers since some development experiments suggested this increased performance slightly.

The retrieval engine was similar to that employed in TREC-8 [12], using the sum of the combined-weights (CW) [20] for each query term to give the score for any given document. For all runs, the value of K used in the CW formula was 1.4, whilst b was set to 0.6 when story boundary information was present (e.g. when using the parallel corpus) or 0 when no document-length normalisation was necessary (e.g. on the windowed test collection). The inclusion of both query and document expansion before the final retrieval stage is discussed in section 4.2.

The final recombination stage pooled all windows which were retrieved for a given query which originated within 4 minutes of each other in the same episode. Only the highest scoring window was retained, with the others being placed in descending order of score at the *bottom* of the ranked list. Although this means that temporally close stories cannot be distinguished, we assume that the probability that two neighbouring stories are distinct but are both relevant to the same query is less than the probability they are from the same story which drifts in and out of relevance. Although alternative, more conservative strategies are also in use (see e.g. [2]), this strategy proved effective in development experiments [15].

4.2. Document and Query Expansion

4.2.1. Query Expansion

Blind Relevance Feedback (BRF) was used to expand the queries prior to the final retrieval stage within our TREC-8 system [12]. The implementation of query expansion used for TREC-9 differs from this in two main ways. The first concerns which index files to use for the expansion, and the second how to weight the query terms after the expansion stage.

In previous work we ran blind relevance feedback first on the parallel corpus only (PBRF), followed by another run on the test corpus alone (BRF) before the final retrieval stage (e.g. [12]). The idea behind this 'double' expansion was to use the larger parallel corpus, which contained knowledge of story boundaries and had no transcription errors, to add *robustly* related terms to the query before running the standard BRF technique on the test collection. Including both stages of BRF was found to be helpful to performance [16]. However, we have found it very sensitive to the number of terms added, t, and number of documents assumed relevant, r, for each stage. Recent work has used a single stage of query expansion on the union of the parallel and test collections (UBRF) before the final retrieval stage [28]. This gives similar results but is less sensitive to the values of t and r chosen and hence was used in the TREC-9 system.

The method of adding and re-weighting terms during query expansion was changed from TREC-8 to follow the specifications given in [25] and [26] more strictly. All terms were ranked using their Offer Weights (OW), but only those which did not occur in the original query were then considered as potential terms for expansion. The final matching score was obtained by using the MS-RW formula as described on page 798 of [26]. Unlike in previous years, both the original terms and the new expanded terms were reweighted using their Relevance Weight (RW).

4.2.2. Document Expansion

Whilst document expansion has been shown to be beneficial for the case where story boundaries are known [22, 23, 28], it does not seem to have been explored for the SU case. We therefore implemented a document expansion stage for our SU windowing system based on that used in our TREC-8 SK system [12], namely:

1. Form a pseudo-query for each window containing more than 10 different terms, consisting of each distinct term
2. Run this pseudo-query on the parallel collection, giving equal weight to all terms
3. Find the top t expansion terms with the highest Offer Weight from the top r documents
4. Add each expansion term to the window once (i.e. increase the term frequency for each expansion term by 1)

Experiments varying the values of t and r showed that the best performance was obtained for $t = 100, r = 15$ for the TREC-8 queries. This document-expanded index file was then used for the final retrieval stage along with the queries generated *before* document expansion.

4.2.3. Results

The results from including query and document expansion within the SU system on TREC-8 queries are summarised in Table 8 and graphically illustrated in Figures 3 and 4.

When there is no query expansion, document expansion increases mean average precision by 25% and 15% relative for short and terse queries respectively. For moderate query expansion (e.g. $t \leq 8$), document expansion is beneficial for both short and terse queries, but this advantage disappears as the level of query expansion increases. Although the best result for the short queries is obtained when including document expansion (51.72% vs 51.53%), the best performance for the terse queries is considerably worse when including document expansion (47.65% vs 50.56%) and thus it was *not* included in the final system.

The values of $t = 20, r = 26$ were chosen for the UBRF stage despite the fact that they were not optimal for either the short or the terse queries, since they provided more consistent performance across the different query sets.

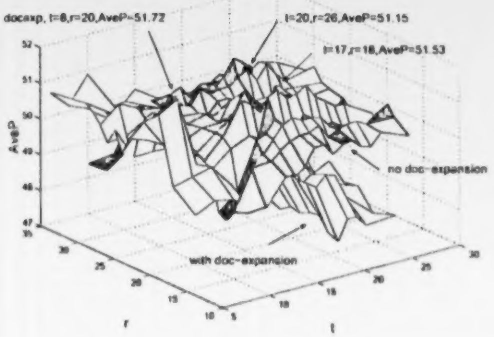


Figure 3: Effect of Query and Document Expansion on TREC-8 *short* queries for SU task on `s1` transcriptions.

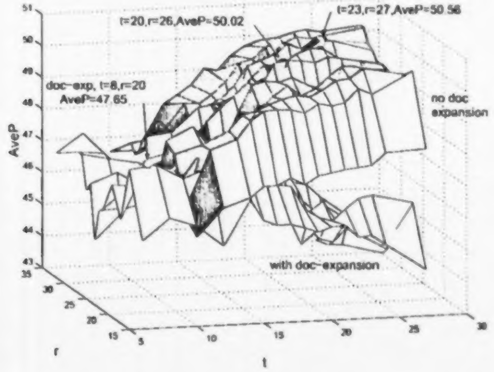


Figure 4: Effect of Query and Document Expansion on TREC-8 *terse* queries for SU task on `s1` transcriptions.

DocExp		QryExp		Short Q		Terse Q	
t	r	t	r	AveP	R-P	AveP	R-P
-	-	-	-	30.89	33.92	32.51	36.77
-	-	8	20	50.84	52.54	47.28	48.43
-	-	17	18	**51.53**	51.78	49.37	49.66
-	-	*20*	*26*	*51.15*	*51.78*	*50.02*	*49.79*
-	-	23	27	51.06	51.60	$\boxed{\mathbf{50.56}}$	50.02
100	15	-	-	38.68	42.42	37.27	41.01
100	15	8	20	$\boxed{\mathbf{51.72}}$	52.61	**47.65**	48.70
100	15	17	18	49.19	49.17	47.00	47.90
100	15	20	26	48.94	49.27	46.67	49.45
100	15	23	27	49.03	49.45	44.48	47.32

Table 8: Interaction of Query and Document Expansion on SU task on `s1` transcriptions.

4.3. Changing the Window Skip

Recent work at Sheffield [19] suggested that increasing the overlap between windows by decreasing the skip during window generation could help improve performance. A contrast run with their lower skip time was thus made to see if this would have helped our system. The results, given in Table 9, show that this would not have been beneficial to our system, which uses a significantly different method of final window recombination to that used in Sheffield's system.

	Short Queries		Terse Queries	
Windowing System	AveP	R-P	AveP	R-P
length 30s, skip 15s	**51.15**	51.77	**50.02**	49.79
length 30s, skip 9s	48.35	50.27	47.25	48.67

Table 9: Effect of reducing the skip size in window generation for `s1` transcriptions for SU TREC-8 queries.

4.4. Summary

Thus to summarise, after our trials with the TREC-8 queries, our TREC-9 SU evaluation system used windowing, filtering of potential commercials, relatively simple indexing, query but not document expansion, standard Okapi weighting and post-retrieval merging. The query expansion was performed on the union of the test and the parallel text collections.

5. THE FINAL TREC-9 SU SYSTEM

The results using the TREC-9 evaluation SU system on all transcriptions are given in Tables 10 and 11 for the (development) TREC-8 and (evaluation) TREC-9 query sets respectively, whilst the relationship between performance and WER is illustrated in Figure 5.

Transcriptions		Short Q.		Terse Q.	
ID	WER	AveP	R-P	AveP	R-P
r1	10.3	51.04	51.86	48.87	50.77
(cuhtk)-s1	20.5	**51.15**	51.78	**50.02**	49.79
cr-limsi2	21.2	50.90	51.07	49.76	50.03
cr-limsi1	21.5	48.75	49.42	47.47	48.09
cr-cuhtk99p1	26.6	49.34	50.92	47.18	47.88
b1	26.7	48.08	48.92	48.17	48.89
cr-nist99b1	27.5	48.37	49.05	47.86	48.36
cr-shef2	29.2	48.30	50.42	47.69	47.45
cr-shef1	32.0	46.91	48.75	46.55	47.38

Table 10: Cross-recogniser results for (development) TREC-8 queries using the TREC-9 SU evaluation system.

The results confirm the conclusions from earlier work in SDR [8], that the decline in performance as WER increases is fairly gentle (-0.17%AveP/%WER on average here). The relative degradation with WER for the TREC-9 and TREC-8 short queries is almost identical (-0.21 vs -0.20 %AveP/%WER), showing that this fall-off is not query-set specific[8].

---

[8] TREC-8 terse queries have a slightly different degradation, but were generated in house with different people and restrictions to those for TREC-9.

Transcriptions		Short Q.		Terse Q.	
ID	WER	AveP	R-P	AveP	R-P
r1	10.3	**40.03**	42.09	**44.02**	47.38
(cuhtk)-s1	20.5	38.83	40.36	42.99	45.02
cr-limsi2	21.2	37.24	39.28	41.62	44.12
cr-limsi1	21.5	36.56	38.57	40.19	43.68
cr-cuhtk99p1	26.6	37.26	39.49	40.44	42.92
b1	26.7	37.08	39.91	40.75	43.87
cr-nist99b1	27.5	36.08	39.86	40.99	44.39
cr-shef2	29.2	37.03	39.48	39.83	42.65
cr-shef1	32.0	36.44	38.96	39.58	42.42

Table 11: Cross-recogniser results for the TREC-9 SU eval.

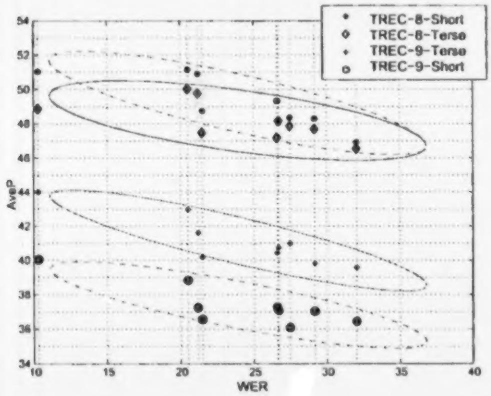


Figure 5: Relationship between WER and AveP for the TREC-9 system on TREC-8 and TREC-9 queries. The ellipses represent 2 standard-deviation points.

The performance on the TREC-8 (development) queries is significantly higher than that on the TREC-9 (evaluation) queries. This may be in part due to three reasons, namely

1. The parameters were tuned for the TREC-8 queries, and may thus be sub-optimal for the TREC-9 queries.
2. All commercials and "filler" portions (e.g. those which summarise stories coming up) were also evaluated for relevance in TREC-9, whereas they were assumed *irrelevant* for TREC-8. Over the 50 TREC-9 queries, there were 93 instances of these portions being scored as relevant. Since our system tries to remove portions such as these by automatically removing commercials before retrieval and biasing the post-processing towards removing fillers[9], the new relevance assessment procedure may have detrimentally affected our score.
3. Natural variation in query difficulty may have meant the TREC-9 queries were "harder" than the TREC-8 ones[10].

To investigate point 2 further, the TREC-9 runs were re-scored using the TREC-8 procedure, which assumed all non-news portions were irrelevant. This increased Average Precision by 1.9% on average for the `b1`, `s1` and `r1` runs for both query sets. This is partly because our SU system tries to filter out the non-news portions before retrieval.

The number of relevant stories from each episode for each query was counted to investigate the validity of the assumption made during post-processing, that the probability of a given episode containing more than one relevant story for a given query was small. The results illustrated in Figure 6 show that 72% of all the relevant stories are unique to their episode and query, but there remains the potential to increase performance by altering the post-processing strategy to allow more temporally close distinct hits[11].

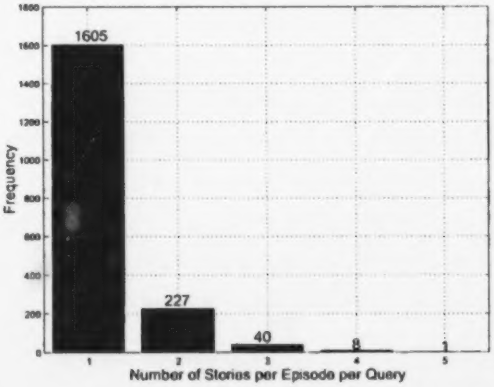


Figure 6: Number of relevant "stories" from each episode for each TREC-9 query .

The expansion parameters were chosen so that the results for the terse and short TREC-8 queries were similar, meaning *sub-optimal values were chosen when considering the short queries alone*. When compared to the (similar) system from Sheffield, whose parameters were chosen based *solely on the short queries*, we do more poorly on average for the short-query runs, but our results are better for all terse query runs [19].

In addition, the parameters $t = 17, r = 18$ which gave the best performance on the short development queries, give better performance on the TREC-9 short queries (AveP=39.38% on `s1`), but worse on the terse queries (AveP=42.78% on `s1`). This suggests that the choice of parameters should take the expected test query length into account and that performance over a wide range of queries might be increased if the expansion parameters were made to be functions of query length.

---

[9] By finding only the most relevant portions within a short temporal span in each episode.

[10] For the `r1` run, we got <10% AveP for 8 TREC-9 short queries, but only 3 TREC-8 short queries.

[11] For example, q153 has 5 relevant "stories" from the episode 19980528_2000_2100_PRI_TWD, with start times: 235/371/810/1594/1711 seconds, but post-processing merging 4-minute portions means a maximum of 3 could be retrieved using this strategy.

6. THE EFFECTS OF USING NON-LEXICAL INFORMATION

As mentioned in section 1.1, non-lexical information automatically derived from the audio could be used within retrieval in the TREC-9 evaluation. Thus, as discussed in section 2, we generated information for `segment`, `gender`, `bandwidth`, `nospeech`, (high-)`energy`, `repeat` and `commercial` tags directly from the audio.

For the SU system we used the `commercial` tags to filter out words thought to have originated in commercial breaks, but we made no use of the other tags. Thus for our required SN contrast run, we ran the SU system without filtering out the commercials[12]. As can be seen from Table 12, as well as reducing the amount of data processing by around 13%, filtering out commercials improved performance by a small, but statistically significant[13] amount on both sets of development queries across all 3 transcriptions (`r1`, `s1`, `b1`). For the TREC-9 evaluation queries only the `s1-terse` and `r1-terse` comparisons were statistically significant[14].

Query Set	Run ID	Time Reject.	Short Q.		Terse Q.	
			AveP	R-P	AveP	R-P
TREC-8	SN-r1	0	50.25	50.95	48.18	49.83
	SN-s1	76.2h	50.77	51.20	49.93	50.12
	SN-b1	0	47.86	48.37	47.96	48.85
TREC-8	SU-r1	65.8h	51.04	51.86	48.87	50.77
	SU-s1	92.5h	51.15	51.78	50.02	49.79
	SU-b1	65.8h	48.08	48.92	48.17	48.89
TREC-9	SN-r1	0	40.54	42.50	44.75	47.03
	SN-s1	76.2h	39.00	40.35	42.65	45.11
	SN-b1	0	37.81	40.44	42.17	44.77
TREC-9	SU-r1	65.8h	40.03	42.09	44.02	47.38
	SU-s1	92.5h	38.83	40.36	42.99	45.02
	SU-b1	65.8h	37.08	39.91	40.75	43.87

Table 12: Effect of automatically removing commercials (SU).

Contrast runs were also performed on the development queries using the less conservative `comm2` system and the manual boundaries derived from the SK case. As can be seen from Table 13 using either of these would have resulted in little difference in performance for our own transcriptions. (none significant at the 2% level.)

Other experiments were run for fun on the TREC-8 queries to see the effect of removing various parts of the audio using the non-lexical information, such as high-energy regions, or particular bandwidth/gender segments. The results are given in Table 13 for the `s1` transcriptions, and plotted in Figure 7. The trend is roughly linear, with the best AveP to time-retained ratio being 0.163%AveP/hr when removing all male speakers, whilst the worst is 0.120%AveP/hr when removing female speakers.

Transcriptions			Short Q.		Terse Q.	
Comm.	Reject	ID	AveP	R-P	AveP	R-P
TREC-9	72.8h	r1	50.82	51.69	48.64	51.08
comm2	96.4h	s1	50.72	51.91	49.79	49.59
	72.8h	b1	48.21	49.25	48.31	49.07
manual	113.9h	r1	51.18	52.75	49.37	51.46
comms	126.6h	s1	50.97	52.07	50.18	50.33
(ndx file)	113.9h	b1	48.28	48.66	49.16	49.71
no loud	111.2h	s1	47.92	49.60	47.29	47.93
no nb	127.4h	s1	46.39	48.52	45.56	46.20
no wb	450.1h	s1	7.69	11.26	8.08	11.29
no male	347.9h	s1	25.25	30.02	25.28	30.64
no female	229.6h	s1	32.59	37.33	31.74	36.69

Table 13: Effect of including non-lexical information for TREC-8 queries. (`s1` reject times include time removed in TREC-8 commercial detection and segmentation stages.)

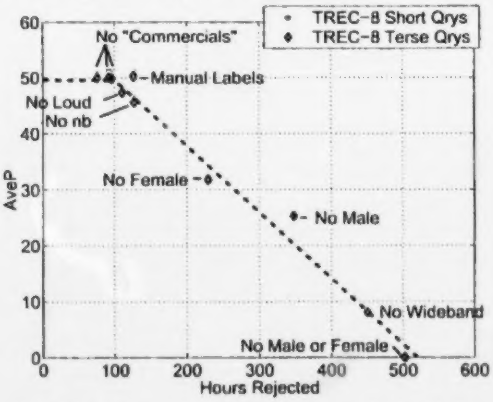


Figure 7: Effect of removing data using non-lexical information on TREC-8 queries for `s1` transcriptions.

7. THE STORY-KNOWN (SK) CONTRAST RUN

The SK system was similar to the SU system described in section 4. The commercial-removal, window-generation and post-merging stages were no longer necessary, since the known story boundaries defined the documents in the collection, but the rest of the system remained practically unaltered.

Document expansion was performed in the same way as described in section 4.2.2 except that the pseudo-query for each document was defined as the 100 terms from the document with the lowest collection frequency. Different values of t and r were investigated for the document expansion stage, but there proved to be little difference between the results, so the values of $t = 200, r = 10$ were chosen to be compatible with [28].

UBRF was performed as described in section 4.2.1, using the *un-expanded* document file to expand the query which was then run on the *expanded* document file, and the values of $b = 0.6$, $k = 1.4$ were retained for all retrieval stages. Results for varying the expansion parameters in the UBRF stage for the SK system are illustrated in Figures 8 and 9 for the short and terse TREC-8 queries and are summarised in Table 14.

---

[12] Note that the `s1` transcriptions already had 76.2hrs of audio filtered out from the TREC-8 segmentation and commercial detection stages [12].

[13] Using the Wilcoxon Matched-Pair Signed-Rank test at the 5% level. (see [16] for discussion of the usage of this test.)

[14] Using the TREC-8 scoring procedure, (non-news portions are assumed irrelevant), *all* TREC-9 SU runs performed better than the corresponding SN runs.

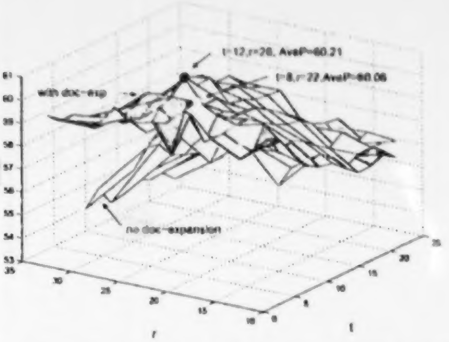


Figure 8: Effect of Query and Document Expansion on TREC-8 *short* queries, SK case, s1 transcriptions.

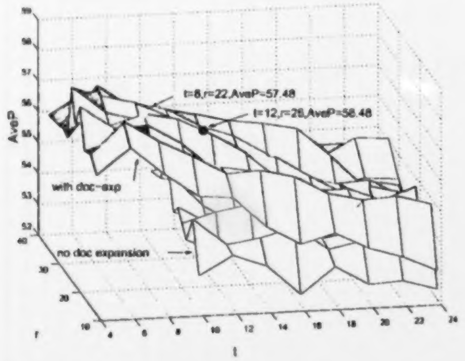


Figure 9: Effect of Query and Document Expansion on TREC-8 *terse* queries, SK case, s1 transcriptions.

The inclusion of document expansion improved performance across both development query sets and all 3 transcriptions, with the largest improvements when the level of query expansion was low to moderate. This consistent improvement was not found for the SU case. The difference is thought to be because the pseudo-queries from windowing for the SU case may be multi-topic, and cannot be as long as for the SK case, since the windows must be kept small (e.g. around 30s) to obtain acceptable performance.

The values of $t = 8, r = 22$ were chosen for the UBRF stage for the SK run to give good performance across both development query sets when used in conjunction with document expansion. The amount of query expansion for the SK case was thus chosen to be less than that used for the SU case because of the interaction between the query and document expansion devices.

The SK results on the TREC-9 evaluation queries are given in Table 15. Since this used a subset of the data and hence also a different relevance file to the SU case, another SK run across *all* the data was performed to allow a more direct comparison between SK and SU cases.

	DocExp		QryExp		Short Q		Terse Q	
Tr.	t	r	t	r	AveP	R-P	AveP	R-P
s1	-	-	-	-	46.29	45.85	45.67	44.53
s1	-	-	8	22	57.41	55.89	54.31	51.31
s1	-	-	12	26	59.11	57.14	54.04	50.65
s1	200	10	-	-	50.76	49.42	52.91	51.67
s1	*200*	*10*	*8*	*22*	*60.06*	*57.62*	***57.48***	*55.15*
s1	200	10	12	26	**60.21**	56.84	56.48	54.88
r1	-	-	-	-	48.19	47.69	47.44	46.28
r1	-	-	8	22	58.17	57.73	54.63	53.19
r1	200	10	-	-	51.65	52.27	53.65	53.76
r1	*200*	*10*	*8*	*22*	*59.04*	*57.31*	*56.95*	*56.20*
b1	-	-	-	-	43.31	43.32	43.17	41.86
b1	-	-	8	22	55.19	54.10	53.04	50.52
b1	200	10	-	-	49.56	48.94	50.86	49.46
b1	*200*	*10*	*8*	*22*	*58.18*	*55.69*	*55.88*	*54.20*

Table 14: Interaction of Query and Document Expansion on SK task for TREC-8 queries.

	Short Queries			Terse Queries		
ID	SK AveP	SK R-P	SU AveP	SK AveP	SK R-P	SU AveP
r1(a)	49.60	47.05	—	52.68	49.26	—
s1(a)	49.47	47.83	—	51.94	50.26	—
b1(a)	48.31	47.38	—	50.44	48.85	—
r1(b)	47.44	45.74	40.04	50.99	48.20	44.02
s1(b)	46.42	44.93	38.83	49.18	48.40	42.99
b1(b)	46.55	46.52	37.08	48.56	47.62	40.75

Table 15: Comparison of TREC-9 SK and SU results. (a) is on the 21,754 story subset, whilst (b) is on all the data, to allow a fairer comparison with the SU case.

Although our SU-SDR system has been improved by around 20% relative[15] since the TREC-8 evaluation [12], and the gap between SK and SU has been reduced from 14% AveP to 8%, there still remains a considerable performance gap between the SK and SU cases.

8. CONCLUSIONS

This paper has described work carried out at Cambridge University for the TREC-9 SDR evaluation. The experiments confirmed that the relative degradation of Average Precision with increasing recogniser error rate is gentle, and performance on high-quality ASR transcriptions can be as good as that on a manually transcribed reference.

Standard indexing techniques and Okapi-weighting provide a good baseline system and adding query expansion using the union of the test and a contemporaneous parallel newswire collection increases performance further. Including a windowing and post-retrieval recombination strategy allows good performance even when no story boundaries are known in advance. Document expansion, which previously has been found to work well for the SK case, was extended to the SU framework and shown to improve performance for small to moderate levels of query expansion.

---

[15] Comparing AveP for s1 on TREC-8 short queries

Non-lexical information derived directly from the audio, which would not normally be transcribed, can be used to improve real SDR systems. Audio repeats can accurately predict the presence of commercials, which can be filtered out before retrieval, and some broadcast structure information can be recovered by analysing cues such as bandwidth, signal energy and the presence of music in the audio. Browsing and understanding could also be improved by including tags such as sentence boundaries and speaker turns. Optimally integrating non-lexical information within real SDR systems, using larger databases and including other information such as video data provide interesting challenges for the future.

Acknowledgements

This work is in part funded by an EPSRC grant reference GR/L49611.

9. REFERENCES

[1] D. Abberley, S. Renals, G. Cook & T. Robinson *Retrieval of Broadcast News Documents with the THISL System*. Proc. TREC-7, pp. 181-190, 1999

[2] D. Abberley, S. Renals, D. Ellis & T. Robinson *The THISL SDR System*. Proc. TREC-8, pp. 699-706, 2000

[3] C. Auzanne, J.S. Garofolo, J.G. Fiscus & W.M Fisher *Automatic Language Model Adaptation for Spoken Document Retrieval*. Proc. RIAO 2000, Content-Based Multimedia Information Access, pp. 132-141, 2000

[4] R. Ekkelenkamp, W. Kraaij & D. van Leeuwen *TNO TREC7 site reports: SDR and filtering*. Proc. TREC-7, pp. 519-526, 1999

[5] M. Franz, J.S. McCarley, R.T. Ward *Ad hoc, Cross-language and Spoken Document Information Retrieval at IBM*. Proc. TREC-8, pp. 391-398, 2000

[6] J.S. Garofolo, J. Lard, C.G.P. Auzanne & E.M. Voorhees 2000 TREC-9 Spoken Document Retrieval (SDR) Track Evaluation Specification.
http://www.nist.gov/speech/tests/sdr/sdr2000/sdr2000.htm

[7] J.S. Garofolo, J. Lard & E.M. Voorhees 2000 TREC-9 Spoken Document Retrieval Track: Overview and Results. To appear in Proc. TREC-9

[8] J.S. Garofolo, C.G.P. Auzanne & E.M. Voorhees *The TREC Spoken Document Retrieval Track: A Success Story*. Proc. RIAO 2000, Content-Based Multimedia Information Access, pp. 1-20, 2000

[9] J.-L. Gauvain, Y. de Kercadio, L. Lamel & G. Adda *The LIMSI SDR System for TREC-8*. Proc. TREC-8, pp. 475-482, 2000

[10] J.-L. Gauvain, L. Lamel, C. Barras, G. Adda & Y. de Kercadio *The LIMSI SDR System for TREC-9*. To appear in Proc. TREC-9

[11] T. Hain, S.E. Johnson, A. Tuerk, P.C. Woodland & S.J. Young *Segment Generation and Clustering in the HTK Broadcast News Transcription System* Proc. DARPA Broadcast News Transcription and Understanding Workshop, pp. 133-137, 1998

[12] S.E. Johnson , P. Jourlin, G.L.Moore, K. Spärck Jones & P.C. Woodland *Spoken Document Retrieval for TREC-8 at Cambridge University*. Proc. TREC-8, pp. 197-206, 2000

[13] S.E. Johnson *Who Spoke When? - Automatic Segmentation and Clustering for Determining Speaker Turns*. Proc. Eurospeech, Vol. 5, pp. 2211-2214, 1999

[14] S.E. Johnson, P.C. Woodland *A Method for Direct Audio Search with Applications to Indexing and Retrieval*. Proc. ICASSP 2000,Vol. 3, pp. 1427-1430, 2000

[15] S.E. Johnson , P. Jourlin, G.L.Moore, K. Spärck Jones & P.C. Woodland *Audio Indexing and Retrieval of Complete Broadcast News Shows*. Proc. RIAO 2000, Content-Based Multimedia Information Access, Vol. 2, pp. 1163-1177, 2000

[16] P. Jourlin, S.E. Johnson , K. Spärck Jones & P.C. Woodland *Spoken Document Representations for Probabilistic Retrieval*. Speech Communication, Vol 32, No. 1-2, Sept. 2000, pp. 21-36

[17] C. Ng, R. Wilkinson & J. Zobel *Experiments in spoken document retrieval using phoneme n-grams* . Speech Communication, Vol 32, No. 1-2, Sept. 2000, pp. 61-77

[18] D. Oard *User Interface Design for Speech-Based Retrieval*. Bulletin of the American Society for Information Science, 26(5) pp. 20-22, June/July 2000

[19] S. Renals & D. Abberley *The THISL SDR system at TREC-9*. To appear in Proc. TREC-9

[20] S.E.Robertson & K.Spärck Jones *Simple, Proven Approaches to Text Retrieval*. Technical Report TR356 Cambridge University Computer Laboratory, May 1997.

[21] E. Scheirer & M. Slaney *Construction and Evaluation of a Robust Multifeature Speech/Music Discriminator*. Proc. ICASSP'97, pp. 1331-1334, 1997

[22] A. Singhal, J. Choi, D. Hindle & D.D. Lewis *AT&T at TREC-7*. Proc. TREC-7, pp. 239-251, 1999

[23] A. Singhal & F. Pereira *Document Expansion for Speech Retrieval*. Proc. SIGIR '99, pp. 34-41, 1999

[24] A. Singhal, S. Abney, M. Bacchiani, M. Collins, D. Hindle & F. Pereira *AT&T at TREC-8*. Proc. TREC-8, pp. 317-330, 2000

[25] K. Spärck Jones, S. Walker, S.E. Robertson *A probabilistic model of information retrieval : Development and status*. Technical report, TR-446, Computer Laboratory, University of Cambridge, 1998.

[26] K. Spärck Jones, S. Walker, S.E. Robertson *A probabilistic model of information retrieval : Development and comparative experiments, Parts 1 and 2*. Information Processing and Management 36(6) pp. 779-840, 2000

[27] A. Tuerk, S.E. Johnson , P. Jourlin, K. Spärck Jones & P.C. Woodland *The Cambridge University Multimedia Document Retrieval Demo System*. Proc. RIAO 2000, Content-Based Multimedia Information Access, Vol. 3, (Applications) pp. 14-15, 2000.

[28] P.C. Woodland, S.E. Johnson, P. Jourlin, & K. Spärck Jones *Effects of Out of Vocabulary Words in Spoken Document Retrieval*. Proc. SIGIR'2000 pp. 372-374, 2000

For TREC publications, see `http://trec.nist.gov/pubs.html`
For Cambridge University SDR papers, see
`http://svr-www.eng.cam.ac.uk/research/projects/mdr/`

kNN at TREC-9

Tom Ault and Yiming Yang {TOMAULT,YIMING}@CS.CMU.EDU
Language Technologies Institute and Computer Science Department
Newell Simon Hall 3612C, Carnegie Mellon University
Pittsburgh, PA 15213-8213, USA

Abstract

We applied a multi-class k-nearest-neighbor based text classification algorithm to the adaptive and batch filtering problems in the TREC-9 filtering track. While our systems performed well in the batch filtering tasks, they did not perform as well in the adaptive filtering tasks, in part because we did not have an adequate mechanism for taking advantage of the relevance feedback information provided by the filtering tasks. Since TREC-9, we have made considerable improvements in our batch filtering results and discovered some serious problems with both the T9P and T9U metrics. In this paper, we discuss these issues and their impact on our filtering results.

1. Introduction

We participated in the TREC-9 information filtering track, submitting results for the OHSU and full MeSH topic sets for the batch and adaptive filtering tasks. We used a filtering engine based on the multi-class kNN algorithm successfully applied to other text categorization problems reported in the literature[6, 5]. In adapting kNN to the TREC-9 filtering tasks, we faced the following challenges:

1. Find the optimal per-category decision thresholds, given that these thresholds vary with the quantity and quality of the training data.
2. Take advantage of the relevance feedback information provided by the TREC filtering tasks.

In our official submissions, we were somewhat successful in meeting the first challenge, and not at all successful in meeting the second. As a result, we did well in the batch filtering tasks, but not as well in the adaptive filtering ones.

Since TREC-9, we have accomplished the following:

1. Increased the performance of our system for the batch filtering tasks by improving our threshold calibration methods and exploring alternative scoring mechanisms.
2. Discovered problems with the T9U and T9P metrics used for official evaluation in TREC-9.
3. Developed an effective mechanism for taking advantage of relevance feedback information.

We discuss the first two accomplishments in this paper. Although our new relevance feedback mechanism is promising, we have just begun to experiment with it, and so discussion of it is deferred to a future work.

This paper has five sections past the introduction. Section 2 describes our filtering system, including the multi-class kNN classifier and our threshold calibration mechanisms; section 3 summarizes our official TREC-9 submissions, while section 4 discusses improvements to our batch filtering results, and in section 5, we analyze the problems inherent in the T9P and T9U metrics and suggest an alternative metric for future evaluations. Section 6 presents our conclusions and future research goals for information filtering.

2. System Description

2.1 Multi-class kNN

We used the multi-class kNN algorithm previously applied by Yang et. al. to the OHSUMED[5] and Reuters collections[6] for our document filtering experiments. We chose this version of the algorithm over the single-class algorithm used in our TDT work because of the large number of categories in the MeSH topic set. Unlike the single-class variants, the multi-class kNN algorithm effectively considers all categories "simultaneously" and is much more efficient for large topic sets.

Documents are represented using the conventional vector space model in which each element is a weighted term corresponding to a token (word) appearing in the T (title), W (abstract), A (author), and S (source, e.g. journal) sections of the document. A document is parsed into a vector of term-weights by breaking the content of the T, W, A and S sections into tokens[1], eliminating stop

---

[1] A token is the longest occuring sequence of alphanumeric

BEST COPY AVAILABLE

words taken from a conventional list, stemming with the Porter stemmer, and computing term weights using a variation of the Okapi term-weighting formula[3, 1]:

$$w(t,\vec{d}) = \frac{tf(t,\vec{d})}{0.5 + 1.5 * \frac{len(\vec{d})}{avg\_len} + tf(t,\vec{d})} \times \frac{log(0.5 + N - n(t))}{0.5 + n(t)} \quad (1)$$

where

$w(t,\vec{d})$ is the weight of term t in document $\vec{d}$;

$tf(t,\vec{d})$ is the within-document frequency of term t;

N is the number of documents in the training set;

$n(t)$ is the number of training documents in which t occurs;

$len(\vec{d})$ is the number of tokens in document $\vec{d}$ after stemming and stop-word removal, e.g. $\sum_{t \in \vec{d}} tf(t,\vec{d})$

avg_len is the average number of tokens per document in the training set, e.g $\frac{1}{N}\sum_{i=1}^{N} len(\vec{d}_i)$

The values of N, $n(t)$, and avg_len were computed from the entire training set, but were not updated as the test set was processed, because dynamic updating of training set parameters slows down our current, soon-to-be-improved, document indexing system.

The basic multi-class kNN algorithm has three steps:

1. Index the training set
2. For each document $\vec{x}$ to be classified, retrieve its k most-similar documents from the training set (where k is a parameter of the algorithm). Call this set $\mathbb{R}_k(\vec{x})$.
3. For each category C, compute its relevance to $\vec{x}$ as:

$$s(C,\vec{x}) = \sum_{\vec{d} \in \mathbb{R}_k(\vec{x},C)} sim(\vec{d},\vec{x}) \quad (2)$$

where $\mathbb{R}_k(\vec{x},C)$ is the subset of documents in $\mathbb{R}_k(\vec{x})$ that are relevant to C.

We use the standard cosine-similarity metric to compute similarity between the training and test documents (e.g. $sim(\vec{d},\vec{x}) = cos(\vec{d},\vec{x}) = \frac{\vec{d}\cdot\vec{x}}{\|\vec{d}\|\|\vec{x}\|}$). If we let

N_{train} be the number of documents in the training set

N_c be the number of categories being evaluated over

$|v|$ be the size of the training set vocabulary

$\bar{v}$ be the average number of words per document

$\bar{c}$ be the average number of categories per document

then step (1) takes $O(N_{train}\bar{v})$ time and space, step (2) takes $O(\frac{N_{train}\bar{v}^2}{|v|})$ time and no additional space, and step (3) takes $O(N_{train}logk) + O(k\bar{c})$ time and $O(k + N_c)$ space. Note that these complexities do not include the time it takes to convert a document to its vector space representation.

There are many ways to transform the scores $s(C,\vec{x})$ for a particular category-document pair into a YES/NO decision on whether to assign that document to that category. In this paper, we consider three methods which have been widely reported in the text categorization literature, which we call SCut, RCut and PCut respectively[6]:

- *SCut*: Assign to each category a threshold $t(C)$. Assign a document $\vec{x}$ to category C if $s(C,\vec{x}) \geq t(C)$. How the category-specific thresholds $t(C)$ are set for the multi-class kNN algorithm is discussed in section 2.2.
- *RCut*: For each document $\vec{x}$ in the evaluation set, sort its scores $s(C,\vec{x})$ in descending order and assign the top R-ranking categories in this list to $\vec{x}$, where $R > 0$ is an integer parameter of the RCut scoring method.
- *PCut*: For each category C, sort the scores $s(C,\vec{x})$ for it in descending order. Assign to C the top $N_P(C)$-ranked documents $\vec{x}$ for that category, where $N_P(C) = N_{doc} \times K \times P(C)$, N_{doc} is the number of documents in the evaluation set, K is a user-specified parameter, and $P(C)$ is the estimated prior probability of category C. In this paper, we tested our system on two different ways to compute $P(C)$:
  - *Uniform*: $P(C) = 1/N_c$
  - *Training Set Relative Frequency*: $P(C) = N_{train}(C)/N_{train}$, where $N_{train}(C)$ is the number of documents in the training set assigned to category C.

  When necessary to distinguish the two variants, we call the former "Uniform PCut" and the latter "Proportional PCut."

The relative strengths and weaknesses of these three scoring methods are investigated in a separate paper[7]. Because the PCut method requires scores to be assigned to all documents in the evaluation set before decisions about category assignments are made, this method is not suitable for use in the TREC-9 filtering track, given its constraint that category assignment decisions must be made on-line. We present results for this scoring method only for comparison purposes with the other two methods, RCut and SCut, both of which can make category assignment decisions in real-time.

2.2 Parameter Calibration

Before the multi-class kNN algorithm can be used, the value of k must be set. We used standard m-way cross-validation to set this value; the OHSUMED-87 training data was split into m partitions, with documents assigned randomly to each partition. For each cross-validation run, m_{tr} of these partitions formed the training subset and $m_{va}(= m - m_{tr})$ partitions the validation subset. Partitions were rotated between the training and validation subsets so that each partition was used m_{tr} times for training and m_{va} times for validation. Performance was averaged over all m runs to produce a final value used for comparison between different values of k. In our experiments, we considered $k = 10, 50, 100, 200$ and settled on $k = 200$.

---

characters, dashes ("-"), or underscores ("_") followed by an optional 's or 't digraph)

BEST COPY AVAILABLE

Setting the values of $t(C)$ for the SCut method is a little more tricky, since these values depend on the number and diversity of examples for each category in the training set, as well as many other factors. For our TREC-9 experiments, we explored four different methods for computing $t(C)$:

1. *Through Standard Cross-Validation*
   Use the same m-way cross-validation procedure used to set k, and average the per-category optimal thresholds obtained from each of the m cross-validation runs. If the ratio of m_{tr} to m_{va} is large enough, then the training subset used for each cross-validation run should be sufficiently representative of the complete training data that the averaged thresholds will be sufficiently close to the true optimal values.

2. *Through Linear Regression with Respect to Training Set Size*
   Perform m-way cross-validation with at least three different ratios of m_{tr} to m. For each category, fit a straight line using linear regression to the (ratio, optimal threshold) pairs for that category, and use this straight line to predict the optimal threshold for the full OHSUMED-87 training set. If the thresholds for a category are not sufficiently linear, use the threshold from the largest ratio as a fallback value.

3. *Through Linear Regression with Respect to Number of Examples*
   Perform m-way cross-validation with at least three different ratios of m_{tr} to m, and record the number of examples of each category in the training subset for each cross-validation run. Fit a straight line via linear regression to the $N_{ratios} \times m$ data points for each category, and use the straight line to predict the optimal threshold for a category as a function of the number of examples of that category in the complete training set.

4. *Through a Modified Leave-One-Out Cross-Validation Algorithm*
   Perform a variation of leave-one-out cross-validation on the training data. For each document $\vec{d}$ in the training set, use every other document in the training set to assign scores $s(C, \vec{d})$ via the multi-class kNN algorithm. Then set the values of $t(C)$ to be those which produce optimal performance over this set of scores. This method has the advantage of deriving the values of $t(C)$ from a data set that is as close as possible to the actual training data.

Only thresholding methods (1) and (2) had been developed when the official submissions for the TREC-9 filtering track were due, and hence our official submissions presented in section 3 reflect the performance of these two methods. Thresholding methods (3) and (4) were developed after the TREC-9 submission deadline to improve the performance of our batch filtering systems, and their impact is discussed in section 4.

The values of R for the RCut scoring method and K for the PCut scoring method can also be set using either method (1) or method (4). Because of time constraints, we use method (4) to set these values in this paper, and defer comparison of the two methods to a later work.

3. Official TREC-9 Submissions

We submitted six runs for the TREC-9 filtering track, four *baseline* and two *combination* runs. The baseline submissions (runs CMUCAT1-4 in Table 1) represent our best attempt at tuning the parameters of our basic system, while the combination runs (CMUCAT5 and CMUCAT6 in Table 1) are an attempt to improve performance on the OHSU query set by using a weighted linear combination of the output of the multi-class kNN algorithm on two different views of the documents in the training and evaluation sets, one view in which the abstracts were left unmodified, and another view in which the abstracts had been replaced with the definitions of the MeSH subject headings appearing in the .M section of the document. All runs used $k = 200$ and the SCut scoring method, but *none* of them, including those submitted for the adaptive filtering task, made use of relevance feedback information; only the initial training data was used to filter documents.

The following summarizes our official results

- *Baseline Batch Filtering*
  Thresholds for the baseline batch filtering runs (CMUCAT1 and CMUCAT4) were set using method (2). The poor performance of the baseline run on the OHSU queries (CMUCAT4) can be explained by the lack of linearity with training set size of the thresholds for these categories. Of all 63 categories in this set, only 11 were sufficiently linear with training set size to predict a good threshold for the full training set. In contrast, 3790 (77%) of the 4904 MeSH topics were sufficiently linear with training set size to predict a good threshold. If we use method (1) with a $m_{tr}:m_{va}$ ratio of 19:1, T9P for CMUCAT4 leaps to 0.241, which is comparable to the scores for the other runs.

- *Baseline Adaptive Filtering Runs*
  Thresholds for the adaptive filtering runs (CMUCAT2, CMUCAT3, and CMUCAT5) were set using a version method (1) modified to account for the limited available training data. This variation always keeps the adaptive filtering training documents in the training subset and rotates the remainder of the OHSUMED-87 data set through the validation subset.

- *Comparison of Baseline Runs*
  Multi-class kNN performed better on the batch filtering task than on the adaptive filtering one, which is expected since the former has more training data than the latter. The algorithm also performed much better on the MeSH topics than on the OHSU queries. This may also be because of the larger number of training documents on average per category for the MeSH topics than for the OHSU queries for both tasks (an average of 236.82 documents/topic for MeSH vs. 50.87 documents/topic for OHSU for the batch filtering task, and 4 documents/topic for MeSH vs. 2 documents/topic for OHSU for the adaptive filtering task).

- *Combination Runs* Combining scores from classifying different views of a document seems to improve performance by 1-2% in T9P over the corresponding baseline run (when appropriate thresholding is used).

4. Improved Batch Filtering Performance

4.1 Improved Threshold Calibration For SCut

Increases in the performance of our batch filtering methods come from the development of improved threshold calibration methods for the SCut method and the application of alternative transformations (RCut and PCut) from

BEST COPY AVAILABLE

Table 1: Official Submissions by CMU-CAT for the TREC-9 Filtering Track

Run ID	Task	Topic Set	Use .M Field?	Single or Combination?	T9P
CMUCAT1	Batch	MeSH	No	Single	0.436
CMUCAT1*	Batch	MeSH-SMP	No	Single	0.443
CMUCAT2	Adaptive	MeSH	No	Single	0.303
CMUCAT2*	Adaptive	MeSH-SMP	No	Single	0.304
CMUCAT3	Adaptive	OHSU	No	Single	0.213
CMUCAT4	Batch	OHSU	No	Single	0.100
CMUCAT5	Adaptive	OHSU	Yes	Combination	0.224
CMUCAT6	Batch	OHSU	Yes	Combination	0.261

*The CMUCAT1 and CMUCAT2 runs were used for both the full and "sampled" MeSH topic sets; the CMU-CAT group did not have separate submissions for these two topic sets.

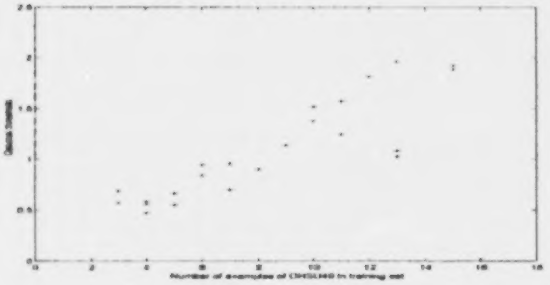


Figure 1. Optimal threshold vs. number of examples in training set for OHSU40

document-category pairs scores to assignment decisions for that category. Figure 1 shows the motivation for threshold calibration method (3). Most, but not all, categories from both the OHSU and MeSH topic sets have optimal thresholds which are linear with the number of examples of that category in the training set. We can use the (number of examples, optimal threshold) pairs gathered from different cross-validation runs to build a linear predictor of the optimal threshold for a category given the number of examples of it currently in the training set.

Threshold calibration method (4) was motivated by the observation that the larger the ratio of m_{tr} to m_{va} used for threshold calibration method (1), the better the thresholds predicted by this method. If we want to classify a document in the training set, the most representative subset of the training data we can use consists of every document in the training set except the one being classified. After scoring every document in the training set in this fashion, we can set the per-category thresholds to be those which yield optimal performance on these scores.

Sometimes this method computes very low thresholds that perform very poorly on both the training and test data. This typically occurs for categories that have few representative examples in the training set, and thus those examples are assigned low scores during the training set self-evaluation. However, since any non-zero T9P, no matter how small, is better than than a zero T9P, method (4) will set the "optimal" threshold for any such category to the score of its highest-ranked positive example, even though that threshold will recall far too many false-alarms when applied to the test data.

To compensate for this behavior, we have added fallback values to threshold calibration method (4). If the performance of the "optimal" threshold computed by method (4) for a category over its set of scores falls below a specified value, then we use the score of the N_{fb}-th ranked document for that category as its threshold instead. How N_{fb} is computed depends on the fallback method chosen. We examined three fallback methods in our post-TREC-9 filtering work: *FBR*, *FBP* and *FBPcut*.

1. *FBR*: $N_{fb}(C) = y$, where y is a constant rank specified by the user for all categories. If y exceeds the number of documents with scores for a category, the score of the lowest-ranked document is used. This method is also called "fallback to constant rank."

2. *FBP*: $N_{fb}(C) = p * N_{doc}(C)$, where $N_{doc}(C)$ is the number of documents in the training set with scores for category C and p is a proportion between 0 and 1 assigned by the user. This method is also called "fallback to proportional rank."

3. *FBPCut*: $N_{fb}(C) = K * N_{doc} * P(C)$ (e.g. the same forumula used for the PCut scoring method), where N_{doc} is the number of documents in the training set, K is a nonnegative user-specified constant, and $P(C)$ is one of the two distributions (uniform or proportional) used by the PCut method in section 2.1. This method is also called "PCut fallback."

Table 2 shows the potential gains possible from adding fallbacks to method (4). For both topic sets, FBPcut and FBR made similar gains in performance while FBP showed no improvement over the no fallback condition. Note that the parameters are the ones that produce optimal performance on the *test* data rather than the training data, and thus represent potential rather than actual gains. The parameters for FBP, FBR a[illegible]BPCut are sensitive to overfitting, and we are exploring effective ways to set them from the training data.

4.2 Alternative scoring methods RCut and PCut

Until recently, we believed that SCut was the top-performing scoring method regardless of the corpus or evaluation conditions. Recent work by Yang[7] has shown that this is not

BEST COPY AVAILABLE

Table 2: Comparison of Different Fallback Methods

Method	Topic Set	Optimal Parameter(s)	T9P
FBP	OHSU	p= 0.05	0.278
FBR	OHSU	y= 10	0.313
FBPCut	OHSU	P= 1.1	0.305
FBP	MeSH	p= 0.05	0.462
FBR	MeSH	y= 10	0.475
FBPCut	MeSH	P= 4,TrainRF	0.474

true. Of the three common scoring methods (PCut, RCut and SCut), which one is better varies with the corpus and the desired ability to make trade-offs between recall and precision. Consequently, we applied the RCut and PCut scoring methods to our TREC-9 batch filtering results to see if they would produce better results than SCut. The results are shown in Table 3.

4.3 Analysis

Except for RCut, all of the methods we explored in this section outperformed SCut with threshold calibration method (2), which we used for our official batch filtering submissions. RCut performed poorly because of its inability fine-tune the number of assignments made by the system, forcing it to draw too many false-alarms or not enough correct documents. PCut performed extrordinarily well, outperforming our best SCut thresholding strategy (method 4) by 7% for the OHSU queries![2] Unfortunately, as mentioned in section 2.1, PCut cannot be used to make real-time assignments.

For the OHSU queries, threshold calibration methods (3) and (4) give equal performance, even though they use very different means to compute the set of optimal thresholds. This suggests that we have found the best possible thresholds from the training data for categories with good performance, and we need to concentrate our efforts on low-performing categories. The improvements in T9P from using fallback thresholds with method (4) support this claim. On the other hand, there is slight but significant improvement between methods (3) and (4) for the MeSH categories, suggesting that further improvement in threshold optimization is possible even for well-performing categories in this topic set.

5. Problems with T9P and T9U

Systems participating in the TREC-9 filtering track were evaluated by one or both of two measures, T9P or T9U, which are defined for a category as:

$$T9P = \frac{A}{max(A+B, \alpha)} \tag{3}$$

$$T9U = max(2A - B, minU) \tag{4}$$

where

> A is the number of relevant documents assigned to that category
>
> B is the number of false-alarms for that category
>
> α is a constant parameter indicating the desired number of documents to be retrieved for that category. Systems which retrieve less than α documents for the category are penalized by having the remaining $\alpha - (A + B)$ documents considered to be false-alarms.
>
> $minU$ is a constant parameter representing the minimum value of the T9U measure. This is to keep large negative utility scores from dominating the system-wide average of T9U.

For TREC-9, α was fixed at 50 for all categories, and $minU$ was fixed at -100 for all categories in the OHSU query set and at -400 for all categories in the MeSH topic set. The overall value of T9P or T9U for a filtering system is the unweighted average of its T9P or T9U values for the individual categories (also known as the *macro-average* of T9P or T9U in the information retrieval literature).

Both T9P and T9U can be rewritten in terms of the number of relevant documents for the category (N_+), and the widely-known information retrieval metrics recall[3] (r) and precision[4] (p):

$$T9P = \begin{cases} p & \text{if } r >= \frac{\alpha}{N_+}p \\ \frac{N_+}{\alpha}r & \text{if } r < \frac{\alpha}{N_+}p \end{cases} \tag{5}$$

$$T9U = \begin{cases} min(N_+ r(3 - \frac{1}{p}), minU) & \text{if } p > 0 \\ \text{Some value in } [minU, 0) & \text{if } p = 0 \end{cases} \tag{6}$$

(Note that if $p = 0$, $A = 0$ and the value of B becomes unrecoverable; hence, in this case, T9U will have some negative value not directly computable from r, p and N_+)

5.1 T9P

Several problems with T9P are immediately visible from an examination of equation 5 and the isocurves of T9P plotted in figures 2 through 4 for $\alpha < N_+$, $\alpha = N_+$ and $\alpha > N_+$ respectively. Specifically:

- In spite of its name, T9P actually measures *recall* if $r < \frac{\alpha}{N_+}p$, and thus the macro-average of T9P is actually a mix of recall and precision values, depending

---

[2] A program bug discovered at the last minute prevented us from evaluating the MeSH topic set using the PCut method.

[3] Recall is defined for a category as the ratio of relevant documents assigned to the category to the number of relevant documents for that category, e.g. A/N_+.

[4] Precision is defined for a category as the ratio of relevant documents assigned to the category to the total number of documents assigned to that category, e.g. $A/(A+B)$

Table 3: Improved Batch Filtering Results for TREC-9

	Offical	Method (3)	Method (4)		PCut	
Topic Set	T9P	T9P	T9P	RCut	Priors	T9P
OHSU	0.100	0.277	0.278	0.013	Training	0.348
MeSH	0.436	0.441	0.463	0.292	Training	*

*An unfortunate program bug that kept us from evaluating the MeSH topics was discovered the night the paper was due.

on where the operating point of each category lies in recall-precision space!

- The isocurves of T9P impose a harsh tradeoff between recall and precision. When the operating point lies above the line $r = \frac{\alpha}{N_+}p$, only improvements in precision will be of any benefit. When below this line, only improvements in recall will have any effect.

- From figure 2, if $\alpha < N_+$, then T9P imposes an effective maximum recall of $\frac{N_+}{\alpha}$. Increasing recall past this point will be of no benefit to the system.

- Likewise, from figure 4, if $\alpha > N_+$, then T9P imposes an effective maximum precision of $\frac{N_+}{\alpha}$. Furthermore, this value is also the maximum value of T9P for the category, and so the macro-average T9P of a perfect filtering system over a set that includes categories with this property will not be 1.0, but some value possibly much less. For example, the full MeSH topic set has a maximum macro-average T9P of 0.99, but the OHSU query set has a maximum of 0.73! This has strong implications for comparisons of macro-average T9P across topic sets.

- The properties of T9P are heavily dependent on the number of relevant documents for a category and thus are not consistent from category to category. This makes the macro-average of T9P an especially confusing and counter-intuitive metric.

Most of the problems with T9P come from the harsh, discontinuous trade-off between recall and precision that occurs at the line $r = \frac{\alpha}{N_+}p$. A metric with a smoother trade-off would not have these problems. While the other metric used for TREC-9, T9U does have a smooth trade-off between recall and precision, it has other problems, which we discuss in the next section.

5.2 T9U

Figure 5 shows the isocurves for the T9U metric. Almost immediately, one can see that T9U is a better metric than T9P because its isocurves have a smooth continuous trade-off between recall and precision. Nor does T9U become insenstive to changes in recall or precision except in two regions:

- When p= 1/3, T9U becomes insensitive to changes in recall because for every correct document assigned, two false-alarms are also assigned.

- In the region on or above the curve $r = (\frac{minU}{N_+})(\frac{p}{3p-1})$ (if $minU < 0$) or in the region on or below that curve (if $minU > 0$), $T9U = minU$ regardless of the value of recall or precision.

However, T9U is not without its problems, most of which are discussed in the final report for the TREC-8 filtering track[2] and summarized here:

- The minimum value of T9U ($minU$) is arbitrary value reflecting a particular user's tolerance of poor performance by the system.

- The maximum value of T9U depends on N_+, the number of relevant documents for a topic, which presents a problem for macro-averaging across topics, since topics with many relevant documents will dominate the average. While T9U could be scaled to fall within the same range for all topics, such scaling has problems of its own, which are discussed in [2].

- In the region where T9U is negative ($p < 1/3$), an increase in recall actually results in a *decrease* in T9U, making it possible for one system to have a higher T9U than a second system which has higher recall and precision.

- A system which recalls *no* relevant documents (or any documents at all) may have a higher T9U than a system that recalls a few correct documents and many false-alarms. However, if we assume that the user ignores any category for which too many false-alarms are returned, then a system which returns a few relevant documents among many irrelevant documents, a system which returns no relevant documents, and a system which returns no documents at all are equally "useful" to the user, and T9U should reflect this with an equal value for for all three systems.

While one might attempt to avoid the problems with T9U by setting $minU = 0$ and thus avoiding the entire region of negative utility where most of the problems occur, this would have the undesirable side-effect of obscuring large performance differences between systems. What is needed is a metric which has the same smooth trade-off between recall and precision in its isocurves that T9U has, but without the undesirable behavior exhibited by T9U when it becomes negative. We examine such a metric in the next section.

5.3 A Proposed Alternative To T9P and T9U

Figure 6 shows the isocurves for van Rijsbergen's F_β-measure [4]

$$F_\beta = \frac{(\beta^2 + 1)pr}{\beta^2 p + r}$$

with $\beta = 1$. Like T9U, F_β's isocurves have a smooth, continuous trade-off between recall and precision. However, F_β has several nice properties not found in T9P or T9U, specifically:

- F_β exhibits its smooth, continuous trade off throughout the entire recall-precision space, never becoming completely insensitive to changes in recall or precision.

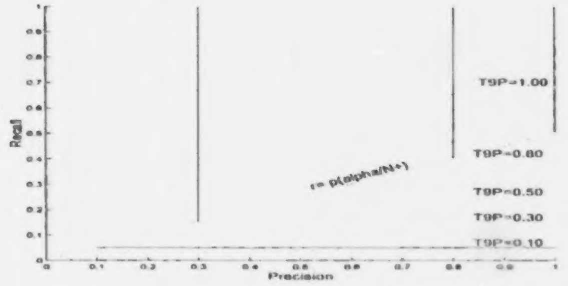


Figure 2. Isocurves of T9P for $alpha/N_+ < 1$

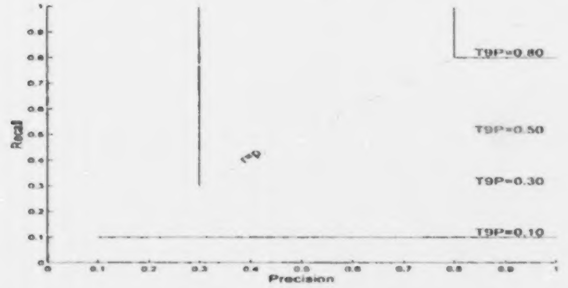


Figure 3. Isocurves of T9P for $alpha/N_+ = 1$

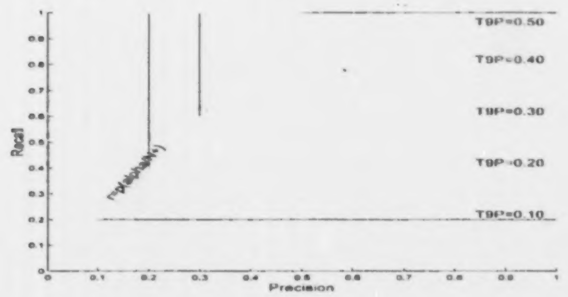


Figure 4. Isocurves of T9P for $alpha/N_+ > 1$

- By adjusting the value of β, one can adjust the recall-precision trade-off of the isocurves to favor recall ($\beta > 1$) or precision ($\beta < 1$). Moreover, this trade- off will be the same for all categories, regardless of the number of relevant documents or other category-dependent properties.

- Expanding on the above item, the range and interpretation of F_β have no category-dependent properties, and thus there are no problems in interpreting its macro-average value.

- In contrast to T9U, systems which return no documents and systems which return only irrelevant documents both have the minimum F_β of zero.

- F_β with $\beta = 1$ is a commonly used metric in information retrieval research, and thus the use of the F_β measure would make the results of the TREC filtering tracks more comparable to other published results in the information retrieval literature.

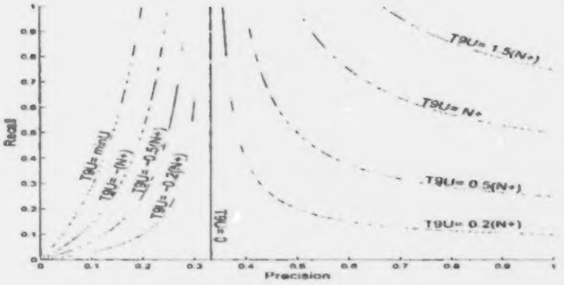


Figure 5. Isocurves of T9U

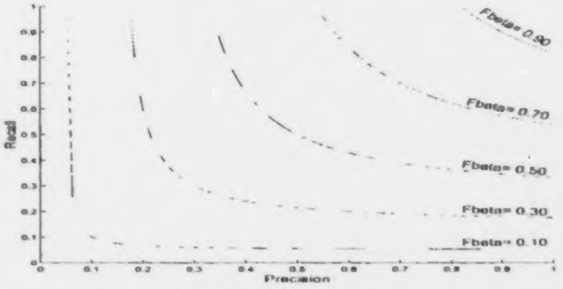


Figure 6. Isocurves of F_β with $\beta = 1$

6. Conclusions and Future Work

We have made considerable progress towards applying the multi-class kNN algorithm to the TREC batch and adaptive filtering tasks. We have developed improved thresholding methods and a promising relevance feedback mechanism. We have also discovered serious problems with the T9P and T9U metrics, and proposed the use of the F_{beta} metric instead. Our efforts in the coming year will focus on exploring the properties of our new relevance feedback mechanism and discovering new ways to tune paramters for our fallback thresholds.

References

[1] M. Franz and S. Roukos. Trec-6 ad-hoc retrieval. In *Proceedings of the Sixth Text REtrieval Conference (TREC-6)*, 1994.

[2] David A. Hull and Stephen Robertson. The trec-8 filtering track final report. In D.K. Harmon, editor, *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, 1999.

[3] S. E. Robertson, S. Walker, S. Jones, M.M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In D.K. Harmon, editor, *Proceedings of the Third Text REtrieval Conference (TREC-3)*, 1994.

[4] C.J. van Rijsbergen. *Information Retrieval.* Butterworths, London, 1979.

[5] Y. Yang. An evaluation of statistical approach to text categorization. In *Technical Report CMU-CS-97-127, Computer Science Department, Carnegie Mellon University*, 1997.

BEST COPY AVAILABLE

[6] Y. Yang. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*, 1(1/2):67–88, 1999.

[7] Yiming Yang. A study on thresholding strategies for text categorization. In *The Twenty-Fourth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'01)*, New York, (submitted). The Association for Computing Machinery.

YFilter at TREC-9

Yi Zhang Jamie Callan
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15232, USA
{yiz,callan}@cs.cmu.edu

ABSTRACT

We built a filtering system YFILTER this year, which we used for experiments on profile updating and thresholds setting. Our focus is using incremental Rocchio for introducing new query terms and term weighting. Although 1, 0.5, 0.25 is a widely used Rocchio ratio for query expansion based on relevance feedback, we found that the optimal setting for information filtering is corpus and profile dependent. In addition to a new Rocchio ratio, we tested a modified idf measure for term weighting (ydf) that is biased towards words with middle range term frequency.

Keywords
Information Filtering

1. INTRODUCTION

Given an initial description of a profile, a filtering system must sift through a stream of information and deliver the most relevant documents to the profile [19]. Filtering is more like a classification problem than a traditional search problem, because of the threshold, which makes it a binary decision process. Many text classification algorithms, such as SVM, Rocchio, Boosting and Naive Bayes, can be applied to filtering [1, 4, 5, 6, 13, 15...], especially for batch filtering and routing. However, as documents arrive sequentially over time, it is unrealistic to use time-consuming algorithms for online filtering. Our goal is to use a computation and storage efficient algorithm, thus making adaptive filtering possible in a normal environment, such as a PC, while still maintaining reasonably good accuracy. Our research interests caused us to adopt stricter constraints than those imposed by the Filtering task [19]. Our filtering system examines each document just once, accruing a small amount of statistical information in the process. The system does not accumulate or store batches of documents, so it requires only minimal storage and moderate computational resources.

A high performance information filtering system should be effective and efficient. Although this is a general requirement for all engineering systems, this year's 5000 MESH profiles make it a clear requirement. A system must handle a large number of user profiles (such as 5000) and a large volume of information efficiently. Previous experiments on text classification suggest that the performance of most text classification algorithms is relatively similar. We hypothesize this will also be true for information filtering. Our starting point, therefore, is to find an algorithm that can be implemented quickly and then to refine it to perform well. We tried several methods that can be implemented incrementally for profile updating (including Rocchio, mutual information and tf.idf). Based on experiments with TREC-8 and TREC-6 filtering data, we used a refined version of the Rocchio algorithm for our final run on this year's OHSUMED data.

2. SYSTEM DESCRIPTION

In order to achieve our goal, we built a new filtering system, called YFILTER. YFILTER is architecturally similar to InRoute [4] and consists of 3 major modules: YParser, YClipset and YLearner, which we used to support dealing with the input stream, filtering according to the profiles, and learning from relevance feedback, respectively. Different learning algorithms for profile updating can be implemented in the YLearner module.

YFILTER supports both structured Boolean and natural language descriptions of initial profiles. For natural language profiles, this system can automatically update the profile according to user relevance feedback.

YFILTER processes text first by removing useless symbols (such as punctuation, and special characters) and fields (such as the .P field), excluding the 418 highly frequent terms listed in the default INQUERY stop words list [3], and then stemming using the Porter stemmer [16].

3. ALGORITHM FOR TREC-9 FILTERING RUNS

3.1 Initial Profile Setting

In all of our experiments, the initial profile is a list of words from the Title and Description fields of the corresponding TREC topic. The weight of each word is its term frequency in the topic.

```
for each document $d_i$
  for each profile $p_k$
    if $d_i$ is filtered to $p_k$ and has feedback //update profile
      update statistics;
      if feedback is relevant
          add all words in $d_i$ to $p_k$'s candidate term list
          calculate weight of each term in $p_k$'s candidate term list according to the Rocchio formula

          sort according to the weight, put the top words in the profile's word list
      else threshold-= max(maxstep, score($d_i$, $p_k$)-threshold)/sqrt(1+number of feedbacks of $p_k$)
    else //automatically decrease threshold
      if(current delivery ratio $\varphi^{'}$<minimum delivery ratio $\varphi$)
          if(performance($p_k$)>0)decrease_step=(Threshold-0.4)* $\varphi$
          else decrease_step=(Threshold-0.4)*($\varphi-\varphi^{'}$)
          threshold=max(threshold-decrease_step, 0.400)
```

Figure 1: An outline of profile updating algorithm

Given 2 initial relevant documents, we update the profile using the Rocchio algorithm [17], and then use the new profile from these 2 initial documents to score other documents without relevance feedback in OHSUMED87. The initial threshold is set to allow the top $\varphi(M+\delta)$ documents in the training dataset to pass. φ is an estimate of the expected minimum delivery ratio we want to achieve. δ is used to make the initial threshold a little lower, so that we can filter more at the beginning, thus obtain more feedback for learning. We arbitrarily set $\delta = 2$ in our experiment.

3.2 Scoring Method

We used the BM25 tf.idf formula [18] for scoring. Idf is initialized based on OHSUMED87 and updated over time as documents are filtered.

3.3 Profile Updating

YFILTER has profile-specific anytime updating. That is, it updates a profile immediately whenever feedback, positive or negative, is available for that profile (Figure 1).

3.3.1 Threshold Updating

The TREC9 ***T9P*** metric is defined as $R_+ / \max(MinD, (R_+ + N_+))$ [19]. *T9P* demands a minimum number of documents (MinD) be delivered to each profile. MinD is set at 50 documents over the 4-year test period, thus approximately 1 per month. We set our delivery ratio φ accordingly. Each negative feedback increases the threshold. Otherwise, the threshold is always decreasing according to φ and the current profile's performance. The magnitude of an increase to a threshold is limited by *maxstep*, which is empirically set to 0.005, so that a non-relevant document with an extremely high score will not push the threshold too high. Thus, we can avoid undue influence of an outlier. For measuring the performance of a profile, we arbitrarily used the utility F2 used in TREC-8 [9]. If a profile's F2 utility is positive, we regard it as a good profile, and, therefore decrease its threshold comparatively faster (Figure 1). All of the parameters are set based on TREC8 and TREC6 data. The intuition is to filter more documents for good profiles, while keeping the delivery ratio for bad profiles at least meet the requirement set by the *T9P* measure.

For the $T9U = \{2 * R_+ - N_+ \;\; if(2 * R_+ - N_+) > MinU\}$ measure [19], if we filter by estimated probability of relevance based on the score of the current document only, the linear component of *T9U* is equivalent to the retrieval rule:

$$\text{Retrieve if } P(rel \mid score) > 0.33.$$

Unfortunately, the sparsity of positive relevance feedback makes it hard to find the optimal threshold for most of the profiles, especially for online searching while doing information filtering. If we set the threshold to the one that yields the best utility over the accumulated documents, while the current profile is built from the same documents, we expect that the resulting threshold is biased. We prefer the *T9P* measure to the *T9U* measure, because when there are no positive utilities, filtering no documents is usually the best strategy. Our profile updating method is *T9P* oriented, and our submission of a *T9U* oriented run is just a small change of *T9P* optimization to make the minimum delivery ratio φ 50 times smaller and to decrease the threshold if the profile's precision is better than average.

3.3.2 Updating Terms and Term Weights

In all of our experiments, each time a positive relevance feedback arrives (including those in the training data), all words in that document are added to the profile's candidate list of terms. Then the weight of each word in the candidate list is calculated according to the incremental Rocchio formula:

$$Rocchio = \alpha \cdot w_q + \beta \cdot w_{rel} - \gamma \cdot w_{non-rel} \qquad (1)$$

Where

$w_q(t)$: max(term frequency of word t in original topics, 0.5)

$$w_{rel}(t) = \frac{1}{rel\_set(t)+1} \sum_{d \in rel\_set(t)} tf\_bel_{t,d} * ydf_{t,b} \qquad (2)$$

$$ydf_{t,d} = -idf_{t,d} \cdot \log(idf_{t,d}) - (1-idf_{t,d}) \cdot \log(1-idf_{t,d}) \qquad (3)$$

$$idf_{t,d} = \kappa \cdot \log((C_d + 0.5) / df_{t,d}) / \log(C_d + 1.0) \qquad (4)$$

$$tf\_bel_{t,d} = tf_{t,d} / (tf_{t,d} + 0.5 + 1.5 \cdot (dl_d / avg\_dl_d)) \qquad (5)$$

The meanings of the above parameters are:

$tf_{t,d}$: Number of times term t occurs in document d

C_d: Number of documents arrived before document d

dl_d: Length of document d

adv_dl_d: Average length of documents arrived before document d

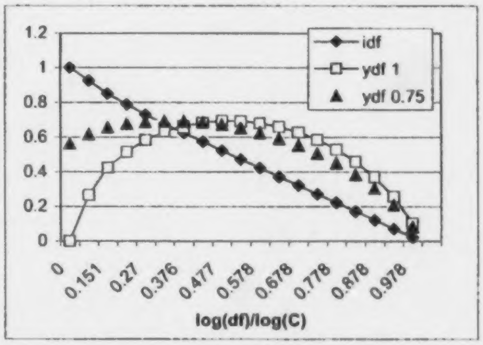
$rel\_set(t)$: Relevant documents after word t is added to the candidate list of the profile

κ: Parameter used to monitoring the query zone

In order to learn faster, we set a bigger β than usual in the relevance feedback formula to emphasize the importance of relevant documents, and changed $w_{rel(t)}$ to emphasize the difference between important words and noisy words.

Some researchers argue that words with middle range term frequencies are more informative than rare words and high frequency words [22], so we introduced a new parameter *ydf* that favors those words. Ideally *ydf* is a convex function that reaches its maximum at the optimal query expansion zone. We arbitrarily calculate *ydf* based on *idf* using Formula 3 & 4. Setting a different κ can move the query zone. Figure 2 shows the effect of setting κ to 0.75 and to 1.0. We first introduced it while testing mutual information, because mutual information favors rare words [23]. We also found it helpful for Rocchio.

Figure 2: Using ydf to favor words with middle range term frequency: the relationship between document frequency and ydf



In order to avoid adding too many noisy words, especially at the early stage, the number of words added to a profile by its t th updating is at most $7/\gamma$. γ increases as the number of relevant documents increases. We also set the maximum number of words for each profile to 60, because our experiments on query expansion for routing task shows that adding more words than that does not improve the performance. Generally speaking, for the number (N_{kt}) of key words in a profile (P_k) after the t th updating, we have:

$$N_{kt} \le \max(N_{k(t-1)} + 7/\lambda, 60)$$

$\lambda = \sqrt{R_+ + 1}$ if current document that triggers profile updating is relevant, otherwise $7/\lambda$ was set to 0. R_+ is the number of relevant documents that have seen for this profile.

4. ANALYSIS OF RESULT

Topic	Optimized for	Average utility/ precision	Percentage of profiles better than or equal to median	Average median utility
MESH	T9P	0.359	0.50	0.41
MESH SAMPLE	T9P	0.363	0.55	0.40
MESH SAMPLE	T9U	26.7	0.53	34.10
OHSU	T9P	0.267	0.68	0.24
OHSU	T9U	9.3	0.97	-3.75
OHSU	T9P	0.279	0.83	0.24
OHSU	T9U	10.1	0.97	-3.75

Table 1: Submitted runs in TREC-9

4.1 Profile Updating

In order to optimize for the measures used for TREC-9, we set the minimum delivery ratio $\varphi = MinD/(6000 * MinD + 50000)$, where 50000 is the approximate number of documents in OHSUMED87. We have not performed a thorough study of the relationship between φ and the actual delivery ratio based on our algorithm. For the $T9P$-oriented run, the delivery ratio is about 1/5000 on OHSU topics, and 1/4400 on MESH topics. This is not surprising, because we are expecting a higher delivery ratio for better profiles (Figure 1). So for MESH topics, which contain more relevant documents on average, the actual delivery ratio is higher than our target. But for the same topic, a higher delivery ratio usually results in lower precision.

In our experiments with TREC-6 and TREC-8, we found that a higher β in the Rocchio formula results in higher performance on both corpora. We guessed that this is also true for other corpora if we have enough number of relevance feedback and the positive feedback itself best represents the corresponding topic. As a result we set $(\alpha, \beta, \gamma) = (1, 3.5, 2)$ for the final run on TREC-9. After submitting our results, we did the same experiment on OHSU topics and the first 50 MESH terms to measure the relation between β and precision. The result confirmed our hypothesis (Figure 3), and we did not observe any decrease on precision as β increases. In fact our setting of the Rocchio ratio was a little conservative, although it is already quite different from the widely used $(1, 0.5, 0.25)$ setting.

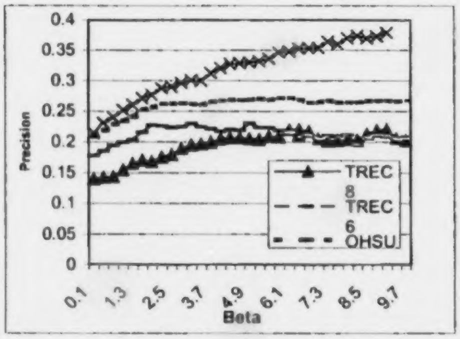


Figure 3: Precision and β setting

4.2 System Performance at Different Stages

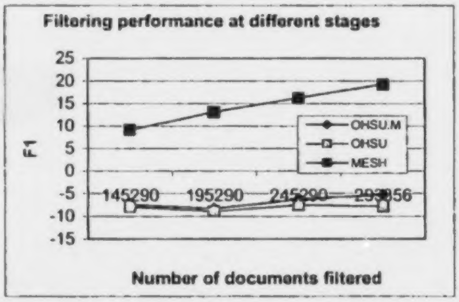


Figure 4: Filtering performances at different stages: F1 Metric

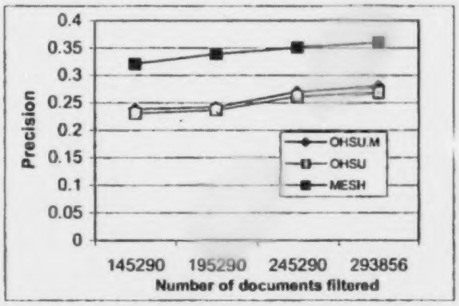


Figure 5: Filtering performances at different stages: Precision Metric

Figure 4 and Figure 5 show our filtering performance using macro average precision and F1 metrics. The horizontal axis does not begin with 0 documents because at the early stage some profiles have no documents filtered to them, and thus the early stage is not comparable with later stages. It is obvious that filtering performance is improving during the whole process in both measurements. This indicates that the filtering system is learning while filtering. Notice that the performance in Figure 4 and Figure 5 is accumulated performance, so the actual snap shot of performance at difference stages is higher than that was showed here.

4.3 Overall Performance in TREC-9

Our results on OHSU are satisfying, while our results on MESH topics are not so good (Table 1). Possible reasons are:

1. Many parameters are set from experiments on TREC-6 and TREC-8. OHSU topics are more like TREC-6 and TREC-8 topics in query length and in number of relevant documents per profile. For example, we set the maximum number of words for each profile to 60, while this should be different from query to query, based on the number of original query terms. Another example is the Rocchio ratio setting. Further experiments show that a higher β setting, such as 9.7, has a very significant improvement on MESH topics (Figure 3). We believe for better performance, β should be set higher.
2. 7 groups submitted OHSU results, while only 4 groups submitted their MESH results.
3. In order to maintain a certain retrieval rate, we introduced a minimum delivery ratio φ for the threshold setting. But the actual delivery ratio is higher than φ, especially for good profiles, where the goodness is measure by the TREC-8 F2 metric. We have more *good* profiles in the MESH runs, so the actual delivery ratio is higher, thus a lower precision.

4.4 Defects and Explanation

Considering that there are too many non-relevant documents for MESH topics, we did not update the profile every time the system encountered a non-relevant document. While the Rocchio accumulator is inside the profile-updating module, thus the filtering system did not accumulate information for non-relevant documents. The effect is equal to $\gamma = 0$ in Formula 1. In fact we want to use 2 for γ based on our experiments on TREC-6 and TREC-8. After submitting the official result, the bug was fixed, which improved recall from 0.363 to 0.376, and improved precision from 0.267 to 0.271 for OHSU. Further experiments show that when β is set bigger, such as 9, the change of γ has no significant impact. One possible explanation is that non-relevant documents are heterogeneous while relevant documents are homogeneous.

5. CONCLUSIONS

We evaluated our new information filtering system YFILTER by participating in the TREC-9 Adaptive Filtering task. It takes about 10 minutes for YFilter to filter 4 years of MEDLINE dataset for 63 OHSU topics. The experimental results compared favorably with results from other filtering systems. In order to maintain a certain retrieval rate, we introduced a minimum delivery ratio φ for threshold setting, and automatically decreased the profile threshold if its delivery ratio is below that. We found the difference between the actual delivery ratio and φ is reasonable on TREC-6, TREC-8 and TREC-9 data, but further theoretical analysis is needed for a more justifiable threshold setting algorithm. We used incremental Rocchio with a quite different Rocchio ratio setting plus a *ydf* measure that favors middle range term frequency words for adding new terms and term weighting. According to further tests on OHSUMED data after submission of our runs, we find that although our new ratio setting has improved the system performance significantly, it is not optimal for TREC-9. Further experiments show that the optimal Rocchio ratio is corpus and profile dependant. For profiles with more relevance judgments, a higher β is much better, so we think a possible solution is to learn β adaptively. As for the TREC-9 run, we found that Rocchio ratios and the φ setting have a significant influence on system performance. We also believe that the idea of introducing *ydf* will improve performance as well, but our function of mapping from *df* to *ydf* is arbitrary, thus the improvement on TREC-9 runs is not obvious. Further experiments can be focused on finding more principled or more accurate functions for *ydf* calculation.

6. ACKNOWLEDGEMENTS

This material is based on work supported by National Science Foundation grant IIS-9873009 and Air Force Research Laboratory contract F30602-98-C-0110. Any opinions, findings, conclusions or recommendations expressed in this paper are the authors', and do not necessarily reflect those of the sponsors.

7. REFERENCE

[1] J. Allan. 1996. Incremental Relevance Feedback for Information Filtering. *Proceedings of SIGIR 1996.*

BEST COPY AVAILABLE

[2] T. A.H. Bell, Alistair Moffat, 1996. In *Proc ACM-SIGIR Conference on Research and Development in Information Retrieval*

[3] J. Broglio, J.P. Callan, W.B. Croft, and D.W. Nachbar, 1995. Document retrieval and routing using the INQUERY system. In *Proceeding of Third Text Retrieval Conference (TREC-3)*, NIST.

[4] J. Callan. 1996. Document Filtering With Inference Networks. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[5] J. Callan. 1998. Learning while Filtering. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*

[6] R.E. Schapire, Y. Singer, A. Singhal. 1998 Boosting and Rocchio Applied to Text Filtering. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*

[7] K. Hoashi, K. Matsumoto, N. Inoue, K. Hashimoto. 1999. Experiments on TheTREC-8 Filtering Track. In *Proceeding of Eighth Text Retrieval Conference (TREC-8)*, NIST.

[8] K. Hoashi, K. Matsumoto, N. Inoue, K. Hashimoto. 2000. Document Filtering Method Using Non-Relevant Information Profile. *Proceedings of SIGIR 2000.*

[9] D A. Hull, S. E. Robertson. 1999. The TREC-8 Filtering Track Final Report. In *Proceeding of eighth Text Retrieval Conference (TREC-8)*, NIST.

[10]D. Hull. 1998. The TREC-7 Filtering Track: Description and Analysis. In *Proceeding of Fourth Text Retrieval Conference (TREC-7)*, NIST.

[11]D.A. Hull. 1997. The TREC-6 Filtering Track: Description and Analysis. In *Proceeding of Sixth Text Retrieval Conference (TREC-6)*, NIST.

[12]T. Joachims 1997. A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. In *Proceedings of International Conference on Machine Learning (ICML).*

[13]Y.H. Kim, S.Y. Hahn, B.T. Zhang, 2000. Text Filtering by Boosting Naive Bayes Classifiers. In *Proceedings of the 23st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*

[14]K.L. Kwok, L. Grunfeld, M. Chan, 1999. TREC-8 Ad-Hoc, Query and Filtering Track Experiments using PIRCS. In *Proceeding of eighth Text Retrieval Conference (TREC-8)*, NIST.

[15] R.D. Lyer, D.D. Lewis, R.E. Schapire, Y. Singer, A. Singhal. Boosting for Document Routing. In *Proceeding of ninth International Conference on Information and Knowledge Management.*

[16]M. F. Porter, 1980. An algorithm for suffix stripping.

[17]J. J. Rocchio. 1971. Relevance feedback in information retrieval in The SMART Retrieval System- Experiments in Automatic Document Processing, pages 313-323. Prentice Hall Inc.

[18]S. E. Robertson, S. Walker, M. M. Beaulieu, and M. Gatford, A. Payne, 1995. A. Okapi at TREC-4. In *Proceeding of Fourth Text Retrieval Conference (TREC-4)*, NIST.

[19]S. E. Robertson, D. A. Hull 2000. Guidelines for The TREC-9 Filtering Track.

[20]A. Singhal, J. Choi, D. Hindle, D. D. Lewis. 1998. AT&T At TREC-7. In Proceeding of Seventh Text Retrieval Conference (TREC-7), NIST.

[21]S. E. Robertson, S. Walker. 1999. Okapi/Keenbow at TREC-8. In *Proceeding of Eighth Text Retrieval Conference (TREC-8)*, NIST.

[22]V. Rijsbergen. 1979 *Information Retrieval*

[23]Y. Yang, J.P. Pedersen 1997. A Comparative Study on Feature Selection in Text Categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97)*

[24]C. Zhai, P. Jansen, E. Stoica.1998. Threshold Calibration in CLARIT Adaptive Filtering. In *Proceeding of seventh Text Retrieval Conference (TREC-7)*, NIST.

[25]C. Zhai, P. Jansen, N. Roma, E. Stoica, D.A. Evans 1999. Optimization in C LARIT Adaptive Filtering. In *Proceeding of eighth Text Retrieval Conference (TREC-8)*, NIST.

TREC9 QA-Track Notebook Paper, NIST, Gaithersburg MD, 2000.

One Search Engine or Two for Question-Answering

John Prager, Eric Brown
IBM T.J. Watson Research Center
Yorktown Heights, N.Y. 10598
jprager/ewb@us.ibm.com

Dragomir R. Radev
University of Michigan
Ann Arbor, MI 48109
radev@umich.edu

Krzysztof Czuba[1]
Carnegie-Mellon University
Pittsburgh, PA 15213
kczuba@cs.cmu.edu

Abstract

We present here a preliminary analysis of the results of our runs in the Question Answering track of TREC9. We have developed a complete system, including our own indexer and search engine, GuruQA, which provides document result lists that our Answer Selection module processes to identify answer fragments. Some TREC participants use a standard set of result lists provided by AT&T's running of the SMART search engine. We wondered how our results would be affected by using the AT&T result sets. For a variety of reasons we could not replace GuruQA's results with SMART's, but we could use document co-occurrence counts to influence our hit-lists. We submitted two runs to NIST for both the 50- and 250-byte cases, one with and one without consideration of the AT&T document result sets. The AT&T document set was only used for a subset of about a third of the questions. This subset exhibited an increase in Mean Reciprocal Answer Rank score of 13% and 8% for the two tasks.

1. Introduction

Question Answering is a computer-based activity that involves searching large quantities of text and understanding both questions and textual passages to the degree necessary to recommend a text fragment as an answer to a question. The TREC Question-Answering track is an attempt to bring together the Information Retrieval (IR) and Natural Language Processing (NLP) or Information Extraction (IE) communities. The strengths of IR lie in the search engines, while those of NLP lie in the ability to parse and analyze text. Indeed, some NLP groups have no expertise or interest in developing their own search engines, yet still wish to participate in the Question-Answering track. To enable these groups to readily participate, AT&T have made available the results of running their version of the SMART search engine (Buckley, 1985; Salton, 1971) on the questions. These datasets consist of the top 50 documents retrieved for each of the questions in the track. These document sets were used by, amongst others, the best-performing entry in TREC8 QA (Srihari and Li, 2000).

The principal advantage of using these hit lists is that a group can concentrate on the information extraction task of finding the answers from a relatively limited quantity of text. (The entire collection contains close to 1 million documents.) A secondary benefit is that all groups who operate this way are on a common footing regarding the IE activity. The principal disadvantage of this approach, of course, is that no custom question (or collection) pre-processing is possible. As a consequence it can happen that no document in the top 50 available contains the answer to a particular question.

Our group has its own search engine, GuruQA, based on Guru (Brown and Chong, 1998), and is thus able to control the entire processing operation, from question processing and text indexing to answer selection. The technique we use, called Predictive Annotation, involves indexing anticipated semantic types, identifying the semantic type of the answer sought by the question, and extracting the best matching entity in answer passages (Prager et al. 2000b). Here we explore the question of whether we are at an advantage or disadvantage by not making use of a respected search engine such as SMART as used by AT&T, and we look at a particular way of gaining the best of both worlds.

---

[1] Work performed while at the IBM T.J. Watson Research Center

2. Background

There is much evidence in the field of text processing that combining the results of a single system acting upon different problem formulations, or of different systems acting on the same queries, provides superior performance over individual systems. Belkin et al. (1993) discuss this in some depth for information retrieval. Amongst others, they cite Saracevic and Cantor (1987) and Turtle and Croft (1991) who demonstrated that combining the results from processing with a single search engine, but with different query formulations of a common information need, would produce increased retrieval performance. Foltz and Dumais (1992) used the same query formulation with multiple search engines, and again found increased retrieval performance.

A similar effect has been shown in other areas of the text-processing field, notably classification/part-of-speech tagging (Brill and Wu, 1998)

Given this history, we suspected that we could gain some improvement in the TREC Question-Answering task by combining the hit lists that our search engine produced with those produced by AT&T, and made available to the track participants. The search engine used by AT&T is the SMART system from Cornell; their internally modified version is described by Singhal (1998).

Unlike our system, described in the next section, the AT&T version of SMART used to generate the document sets for Question-Answering was not tailored to the task. It was the same system they used for participation in the TREC7 "Ad-hoc" task. It uses a standard series of IR techniques, such as stop-word removal, tokenization, rule-based and statistics-based phrase formation, *tfxidf* style term weighting and relevance feedback using Rocchio weights (Rocchio, 1971). It returned a ranked list of documents with no indication of relevant passages within the document.

3. Our System

Our Question-Answering system employs the technique of Predictive Annotation, introduced and described in (Prager et al. 2000a). The technique revolves around the concept of semantic class labels which we call QA-Tokens, corresponding loosely to some of the Basic Categories of Rosch et al. (1976). These are used not only as Named Entity descriptors, but are actual tokens processed by the indexer. The basic operation of our system is as follows.

The question is analysed and the desired answer type is determined. The "wh-words" are replaced by the corresponding QA-Token or set of QA-Tokens (thus "how hot" is replaced by TEMPERATURE$, "when" is replaced by @syn(DATE$, TIME$, YEAR$)). The QA-Tokens identified in the documents are indexed as if they were regular terms. A set of about 400 patterns is used for this conversion. We found in TREC8 that questions that failed to match this way were usually of the form "What X" where X was a relatively rare noun (e.g. "What debts did the Qintex group leave?"). For these cases we used WordNet (Miller, 1995) to find a hypernym synset of X that corresponded to one of our QA-Tokens. In the case of X= "debts", the synset for "monetary-value" corresponds to MONEY$.

WordNet was also used to generate synonym lists of head nouns in the questions. Word-sense disambiguation was performed by calculating co-occurrence counts, as described by Moldovan and Mihalcea (2000), but using the TREC collection instead of the Web.

Stop-words are removed, inflected terms are reduced to their lemma form, morphological variants that go beyond simple inflection are added as synonyms (thus "moved" -> "move" -> "@syn(move, motion)"). Weights are associated with terms, according to the scheme "QA-Tokens > proper names > common words". This in effect is a simple implementation of *idf* weighting, but applied to the lexical classes of the terms being indexed.

A set of text patterns is associated with each of the approximately 50 QA-Tokens. Before the text collection is indexed, it is processed by Textract (Byrd and Ravin, 1999; Wacholder, Ravin and Choi, 1997) which applies these text patterns; when a match is found the text is annotated with the corresponding QA-Token. The indexer indexes not only the base terms but all annotations too. Thus when the indexer encounters "France", for example, it will also index the tokens PLACE$ and COUNTRY$ at the same location.

Search is not document-oriented but passage-oriented, where a passage is one, two or three sentences. Scoring does not use *tf* but a kind of combination match, where each query term found in the passage contributes its weight to the passage's score, but only once for any number of occurrences. Only one (the "best") passage is returned per document, thus inducing a document ranking.

The top *n* (normally *n*=10) passages returned by the search engine are then processed by the Answer Selection module, Ansel (Radev et al., 2000). Textract is used again to identify all of the named entities, including simple noun phrases, in those passages. The named entities are typed by QA-Token (simple noun phrases being THING$s), and seven features, such as search-engine ranking, distance from beginning of sentence, and presence of QA-Token in query, are calculated for each entity. A linear evaluation function, using weights discovered by a machine-learning algorithm, is used to associate a final score with each named entity. Finally, text fragments of the desired size (50 or 250 bytes for TREC9) centered on the best named entities are generated.

4. The experiment

The QA track permits participants to submit up to two runs in each of the 50- and 250-byte sub-tracks. Last year we had developed two different answer-selection modules, neither of which was clearly better than the other, so we submitted one run using each module. Since then we have combined the best of each module to give us a single answer-selection component, and we were looking for significant experimental variations we could develop to take advantage of the two-run opportunity. In particular we wanted to do more than just submit runs with different parameter settings. Consequently we decided to submit one run ("R", for Regular) using our system alone, and another ("A", for AT&T) with reference to the AT&T document set.

We will call this latter set SET-A. Our approach was simply to use these documents to increase the score of documents on our own hit lists if the documents also occurred in SET-A (per question).

We had arbitrarily set an internal hit-list size of 10; that is, the search engine would return the top 10 documents (passages) which would then be forwarded to the Answer-Selection module (Ansel). The scores from the search engine were in the range of 0-2000 (approximately). For the "A" run, we increased the internal hit-list size to 50. For every document that was also on the SET-A hit-list we increased its score on our hit-list by 10,000, and then sorted. This had the effect of putting all of the documents that occurred on both hit-lists ahead of those on ours alone, but keeping the relative order within the two groups. The top 10 documents were then forwarded to Ansel. Since search-engine score is a factor in Ansel processing, we subtracted off any 10,000s. This meant that the passage scores that Ansel saw were exactly the scores that our search engine had given. The effect of considering SET-A was therefore solely in determining whether a passage would appear in the input to Ansel. It is an open question, that we need to answer experimentally, whether this is the best way to combine the hit-lists, and whether we should give documents that occur on both lists a permanently increased score.

Note that we did not add to our hit-list any documents that occurred in SET-A but were not originally in our list. This was primarily because our search engine returns not only a document list, but for every document on the list the offset and length of the best-matching passage – for use in Answer Selection. This information was not available in SET-A. The secondary reason was that it was unclear (without any theory or extensive experimentation) how to assign scores to such additional documents.

The overall effect of considering SET-A was to give the "A" run a slightly improved score over the base "R" run. Before we look at the numbers in detail, we need to address the question of whether any improved performance might be due to the AT&T SMART search engine being intrinsically "better" than ours, at least as the two search engines were deployed for this exercise. To that end, we compared the search engines' performances in the following way.

In this comparison we do not ask whether the system extracted the right answer from the documents being considered, but solely whether the documents contained the right answer (a necessary but not sufficient condition for ultimate system success). We call such a document a "correct" document.

We had available lists of which documents out of the 50 per question in SET-A were correct, in the sense just defined. These lists were posted to the QA track mailing list by Ken Litkowski (`ken@clres.com`). For each question there was a list of 0 to 50 numbers in the range of 1 to 50, corresponding to the positions in the hit-list of documents that contained a correct answer. Whether a document contained a correct answer or not was determined by the document's association with a correct response in the `qa-judgments` file, made available on the TREC web site (http://trec.nist.gov) after the TREC9 submissions deadline. Note that this document-judging scheme admits of two possible sources of error: 1) errors by human judges in developing the judgments file, and 2) documents in the set which contained valid answers but were never chosen by any participating entry, so were never judged.

We generated a comparable set of correct-document lists for our own search engine; we'll call this set SET-R. We are now able to compare the two search engines.

Search engine comparison

The first measure we calculated was Mean Reciprocal Document Rank (MRDR) of the first correct response, in analogy to the way the answer fragments are judged in the QA-track. For each question, the Reciprocal Document Rank is 1 point if the top document in the hit-list contains a correct answer, ½ if the first doesn't but the second does, all the way down to 1/50 if only the 50th document does, or 0 if no document on the list does. The RDR scores are averaged over all of the 682 questions. (There were originally 693 questions but 11 were discarded by NIST for reasons of ill-definition.)

For both systems, the MRDR value was calculated to be 0.49 – in other words, on average both systems produced a document containing the right answer in the second position. The fact that both systems had identical MRDR scores indicates that any improvement of our overall score is due to the complementary nature of the two search engines, not to the intrinsic superiority of one or the other.

Before we leave the subject of search engine comparison, we look at two more measures. Choosing the "best" size for a hit list is a heuristic for which we have no firm data. Using a small hit list is desirable if there is frequently a correct document in a high position, since the subsequent processing will then have relatively little noise to contend with, and precision will increase. Long hits lists, on the other hand, offer greater recall. Therefore we looked at subsets of the 50-document hit lists (always starting at document #1). We ask for a hit list of size N (1<=N<=50) whether there was a correct document on the list. The results were very similar, but not identical, for the two document sets, as shown in Figure 1.

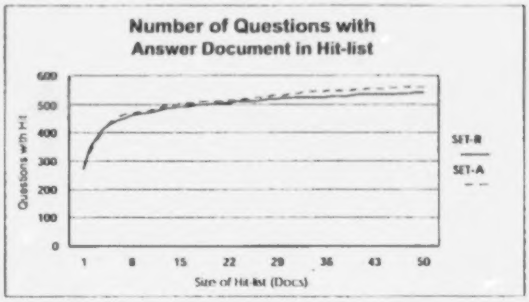


Figure 1. Comparison of number of questions with a "correct" document in the hit list against hit-list depth, for our search engine (SET-R) and AT&T's (SET-A).

The SET-R curve is the solid one, being higher for the first few documents but lower thereafter.

We can make a couple of observations from the data in Figure 1. First, the two search engines seem to have almost equivalent performance. Any significant change in our system's behaviour (good or bad) due to consideration of the AT&T documents will be due to the process of considering multiple result sets, not due to the inherent superiority of one or other set. Second, the curves suggest absolute values for hit list size. To the degree to which the answer-selection processes that operate on these document sets are imperfect, that is, suffer in precision due to the presence of incorrect documents in the set, the hit-lists should be cut back. An obvious "knee" in both curves occurs at around 6-10 documents. On the other hand, if the answer processing is sufficiently sophisticated to be able to easily reject incorrect documents based on their internal semantics, then the hit lists could be quite large. It would be useful then to know at what point the curves, if extended to the right, would asymptote to 100% (=682 questions for TREC9). The only data point we have in this regard is for GuruQA. 542 questions had at least one correct document on a hit-list of size 50; with hit-lists of size 200 we have a correct document for 576 questions. This suggests that the asymptote is far away, and that the correct place to concentrate effort on answering these residual questions is in question pre-processing, and possibly collection processing prior to indexing.

We also looked briefly at the overlap between the document sets. For the entire 50-deep hit lists, we asked which ones contained a correct document somewhere. The totals across 682 documents are presented in Table 1.

# of Questions with correct documents	SET-R Yes	SET-R No
SET-A Yes	483	80
SET-A No	59	60

Table 1. This shows how the two search engines overlapped in answering a question. A document set scores a "Yes" for a question if at least one of the documents in the set contains a strictly correct answer to the question.

It is difficult to make too many quantitative predictions about the potential advantages of using document result sets from multiple systems, since there is more processing to come after the result sets are established.

Actual Performance Increase

We did not attempt to use the AT&T documents for every question, since we did not have a chance to test the idea (using the previous year's sets) before the current year's submission. Instead, we just used them on those question types for which we previously had experienced inferior performance. These were questions, generally of the form "What X ...", for which none of our QA-tokens was instantiated (save for THING$, matching a generic noun phrase, which was created just for such situations).

Of the 682 questions, there were 214 questions which we labelled type THING$. We calculated the Mean Reciprocal Answer Rank (MRAR) score for the THING$, non-THING$ and total sets of questions, both with ("A") and without ("R") the AT&T documents. The MRAR scores reported here are for the actual answers, not the correct documents as MRDR measures in the previous section.

The results are summarized for the 50-byte run in Table 2, and for the 250-byte run in Table 3. Two styles of judging were provided in TREC9: *strict*, in which the answer was present in the returned fragment and justified in the surrounding context, and *lenient*, where the correct answer was present but not necessarily in a context that addressed the question. All of the data reported here were for the *strict* interpretation.

50 byte task	THING$	Non-THING$	Overall
"R"	.151	.390	.315
"A"	.171	.372	.309

Table 2. MRAR scores calculated for THING$, Non-THING$ and all questions, for runs with and without consideration of SET-A documents, for the 50 byte sub-track.

250 byte task	THING$	Non-THING$	Overall
"R"	.335	.454	.416
"A"	.363	.454	.425

Table 3. MRAR scores calculated for THING$, Non-THING$ and all questions, for runs with and without consideration of SET-A documents, for the 250 byte sub-track.

It was expected that there would be a difference between the runs for THING$-type questions, but it can be seen that the Non-THING$ scores also differ between the "A" and "R" runs, in the 50 byte task. This occurred for two reasons, which curiously only affected the Non-THING$ questions. Firstly, there was a word-alignment error in the 50-byte fragment selection code that was present in the "A" system but not in the "R" system. This caused in some cases critical answer words to be truncated and hence disallowed. This affected 6 questions (actually, 2 questions plus 4 paraphrases of one of them), to the tune of a loss of .009 to the MRAR score. The remaining .009 of the discrepancy was due to inconsistent judging (the same answer submitted by both runs was judged differently). Eight questions were negatively affected by these judging problems: in seven cases the "A" run was affected, and in one the "R" run. Unfortunately, the deleterious effects of the inconsistent judging and our alignment bug swamped the positive effect of using the second document set, when the overall scores are calculated (for the 50-byte runs).

From Tables 2 and 3, we see that for the 214 THING$ questions, in the 50-byte sub-track MRAR improved by 13%, and in the 250-byte sub-track by 8%. An experiment that we need to do now is to try a run using the SET-A documents for the Non-THING$ questions too. Due to the labor-intensive nature of the document judging, we will await a set of answer patterns per question from NIST to enable us to judge such future runs automatically.

5. Conclusions and Future Work

Indexing QA-Tokens improves the precision of our IR system, since it gives it more semantics and provides a means of better matching questions to answers. The technique is not so useful in conditions when no semantic type is identifiable, such as "What X" type questions. The experiments reported here demonstrate that considering results sets from a second search engine can improve QA results, at least for those "What X" questions. This was achieved using search engines working under very different operational conditions and with a very basic method of combining the hit-lists. These results extend existing demonstrations of the benefit of using multiple systems in "ad-hoc"-style search and classification.

An incidental discovery was that our use of GuruQA with Predictive Annotation and passage ranking produced result sets with identical MRDR to AT&T's version of SMART, for 214 "What X" type questions. This finding may be related to the existence of theoretical and practical limits to the results achievable with statistical information retrieval.

As mentioned earlier, we plan to see what kind of improvement will be afforded if the approach is extended to all question types. It is also completely open what is the best way to incorporate the information from other result sets. We took the simplest possible approach, which was to move documents that occurred in both hit lists up towards the top of ours. We did not experiment with increasing the score, which we expect will positively effect the results, since Answer-Selection uses passage score as a feature.

References

[1] N.J. Belkin, C. Cool, W.B. Croft and J.P. Callan. "The Effect of Multiple Query Representations on Information Retrieval System Performance", *Proceedings of SIGIR'93*, Pittsburgh, PA, 1993.

[2] E. Brill and J. Wu.. "Classifier combination for improved lexical disambiguation", in *Proceedings of COLING-ACL*, 1998.

[3] E.W. Brown and H.A. Chong. "The Guru System in TREC-6." *Proceedings of TREC6*, Gaithersburg, MD, 1998.

[4] C. Buckley. "Implementation of the SMART information retrieval system." Technical Report TR85-686, Department of Computer Science, Cornell University, Ithaca, NY 14853, May 1985.

[5] R. Byrd and Y. Ravin. "Identifying and Extracting Relations in Text", *Proceedings of NLDB 99*, Klagenfurt, Austria, 1999.

[6] P.W. Foltz and S.T. Dumais. "Personalized information delivery: An analysis of information-filtering methods." *Communications of the ACM*, 35, 12: 51-60, 1992.

[7] G. Miller. "WordNet: A Lexical Database for English", *Communications of the ACM* 38(11) pp 39-41, 1995

[8] D.I. Moldovan and R. Mihalcea. "Using WordNet and Lexical Operators to Improve Internet Searches", *IEEE Internet Computing*, pp. 34-43, Jan-Feb 2000.

[9] J.M. Prager, D.R. Radev, E.W. Brown and A.R. Coden. "The Use of Predictive Annotation for Question-Answering in TREC8", *Proceedings of TREC8*, Gaithersburg, MD., 2000.

[10] J.M. Prager, E.W. Brown, A.R. Coden and D.R. Radev. "Question-Answering by Predictive Annotation", *Proceedings of SIGIR 2000*, pp. 184-191, Athens, Greece, 2000.

[11] D.R. Radev, J.M. Prager and V. Samn. "Ranking Suspected Answers to Natural Language Questions using Predictive Annotation", *Proceedings of ANLP'00*, Seattle, WA, 2000.

[12] J.J. Rocchio. "Relevance feedback in information retrieval" in *The SMART Retrieval System – Experiments in Automatic Document Retrieval*, pp. 313-323, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1971.

[13] E. Rosch et al. "Basic Objects in Natural Categories", *Cognitive Psychology* 8, 382-439, 1976.

[14] G. Salton (ed). *The SMART Retrieval System – Experiments in Automatic Document Retrieval*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1971.

[15] T. Saracevic and P. Kantor, "A study of information seeking and retrieving. III. Searchers, searches, overlap." *Journal of the ASIS*, 39,3: 197-216, 1988.

[16] A. Singhal, J. Choi, D. Hindle, D.D. Lewis, F. Pereira. "AT&T at TREC7", *Proceedings of TREC7*, Gaithersburg, Md., 1999.

[17] R. Srihari and W. Li. "Question Answering Supported by Information Extraction", *Proceedings of TREC8*, Gaithersburg, Md., 2000.

[18] H. Turtle and W.B. Croft. "Evaluation of an inference network-based retrieval model." *ACM Transactions on Information Systems*, 9,3: 187-222, 1991.

[19] N. Wacholder, Y. Ravin and M. Choi. "Disambiguation of Proper Names in Text", *Proceedings of ANLP'97*. Washington, DC, April 1997.

Passive Feedback Collection – An Attempt to Debunk The Myth of Clickthroughs

Christopher C. Vogt
Computer Science, Math and Physics Department
Chapman University, One University Drive
Orange, CA 92866
`vogt@chapman.edu`
`www.chapman.edu/~vogt`

Abstract We report the results of a pilot study designed to investigate the feasibility of collecting information about user actions over the Web. By logging simple events (queries, document views, redisplay of query results) and noting their relative timing, we hoped to be able to predict relevance of viewed documents. Although design problems cast doubt on the accuracy of our results, analysis of the cleanest data reveals that clickthroughs are not very predictive of relevance, but that viewing times, when normalized by document length, are somewhat predictive.

1 INTRODUCTION

Collecting feedback regarding the relevancy of documents from users is an expensive process. The amount of time required can be prohibitive for large collections, even if only the top-ranked documents are scored by humans. Furthermore, in operational settings, users rarely want to be bothered by having to explicitly mark documents as relevant or nonrelevant.

The idea that users select documents for viewing which they think are relevant is an attractive one – it makes determining relevant documents a simple matter of noting "clickthroughs." This idea has gained acceptance in the context of the Web and has also been used in any context where determining relevant documents is difficult [Dreilinger and Howe, 1997].

Our hypothesis is that clickthroughs are actually a poor indicator of relevance, and much too coarse to be of real use. Instead, we suggest that a finer view of the user's actions needs to be used, one which takes into account, amongst other things, the timing of the user's actions. For example, a document which is viewed and then immediately discarded seems much more likely to be *not* relevant. Our experiments in the TREC9 interactive track were designed to make a first step towards investigating the idea that a rich transcript of the actions taken by a user might be used to more accurately predict the relevance of the documents they have viewed.

The most relevant experiment relating to our hypothesis comes from [Morita and Shinoda, 1994]. In this study, users of a newsgroup reader were monitored, and their reading time for each news article recorded. Morita and Shinoda found that the length of time spent reading articles was related to how interesting they were, but not related to their length, their "density," or the amount of backlog (unread articles). Their user task, however, was significantly different from the one used for the interactive track at TREC9. Our subjects were not reading the articles for pleasure, but were actively searching for answers to specific questions. Thus, relevance should play a key role in length of time reading. Furthermore, the reason length of an article played such a small role in the Morita and Shinoda study was that most articles in their study were "uninteresting," so the user spent a very small amount of time on them, regardless of length. The articles that users in our study examined all had a reasonably good chance of being relevant (since the user chooses them based on a headline), so the secondary factor of length could come into play.

2 METHOD

The details of the Interactive Track's experimental protocol are described elsewhere in these proceedings. We describe here only those portions of our experimental setup which were left to individual experimenters to interpret.

2.1 System Designs

As required, we had two different search systems. There is only one difference between our two systems (hereafter cleverly referred to as System 1 and System 2), so we will first describe the common aspects of both systems. We proceeded by indexing each of the 6 information sources (AP, FBIS, FT, LA, SJM, and WSJ) using SMART's [Salton, 1971] "ltc" weighting scheme, no stemming, and

BEST COPY AVAILABLE

the standard stop list. Each of these sources was indexed individually. The results list shown to a user was the result of combining the top 30 documents from each of the 6 sources when the user's query was run against those indexes. System 1 merely sorted these 180 (or fewer) documents according to the RSV (retrieval status value) reported by SMART. System 2 first multiplied the RSV by a weight. This weight was the same for all documents from a given source, but varied by query. The weights were calculated based on half of the subjects' experiences using System 1 in the following manner.

During the first week of the experiment, approximately half of the subjects answered their first four questions using System 1. Using these subjects' answers as a guide, the experimenters assigned all documents that had been viewed to a relevance category (a relevant document being one which directly supported the correct answer to a question). Weights were then assigned to each source on a per-query basis based on the number of relevant (R_s) and nonrelevant (N_s) documents viewed from that source, normalized by the weights assigned to other sources. The initial weight (w_s) given to a source was:

$$w_s = 5R_s - N_s$$

Which was then normalized based on other sources to the final weight, W_s:

$$W_s = 100\left(\frac{w_s - \min}{\max - \min} + 1\right)$$

Where max and min are over w_s from all sources. Table 1 shows the final weights, which were used as multipliers on all documents from a given source on a given question.

2.2 Subjects

The experiment was run entirely over the web. Subjects logged on to a password controlled web site using whatever configuration they had at their disposal. As such, the testing conditions for each subject were not uniform – each used a different computer in a different setting with a different type of internet connection. Users were all volunteers – about half from the greater Chapman community, and half being personal friends of the experimenter. Education level varied from high school diploma through Ph.D. Although over 32 people volunteered, only 30 actually logged any time, and only 25 attempted all 8 questions (with one more attempting 7).

Users were divided into two groups: Group 1 and Group 2. The timeline of the experiment was as follows. During the first week, users in Group 1 completed four questions using System 1. Their results were then used to create System 2, which users in Group 2 used during the second week to answer 4 questions. During the third week, all users answered their remaining 4 questions using the system they had not used for the first four questions. Not all users in all groups adhered to this schedule strictly. Users were allowed to log their sessions at times convenient for them.

2.3 Logging User Actions

One of the main goals of this experiment was to investigate the feasibility of monitoring user actions from afar. To achieve this goal, the system was designed as a collection of CGI scripts (CSH and Perl). User actions would thus be automatically recorded via the HTTP server used to activate these scripts. The the user interface was designed so as to try to maximize the number and kind of events that would be recorded, without interfering too much with usability. It was hoped that the relevance of a document could be inferred from the pattern of browsing events.

2.4 User Interface

An example of what the user interface looked like can be seen in the Appendices. The system divided the Web browser's working area into two frames. The upper, smaller frame contained a textbox for the user's query and a search button. It also contained a textbox for the user's answer, a drop-down list for answer certainty indication, and a Submit Answer button. The question currently under consideration was also displayed in this frame. This upper frame was always visible during the search process. The lower, larger frame served a dual purpose:

1. It displayed the list of results of a search, which consisted of a sorted list of document IDs (in red) and headlines (hyperlinks), along with how many times each word in the query appeared in that document. This word count was included to make scanning though the list easier (and to let the user do a sort of visual version of AND and OR, which were not part of the query language).

2. When the user requested the full text of a document (by clicking on its headline), this frame would then display the full contents. In this configuration, the lower frame was divided vertically into a left and right subframes. The left subframe was narrow, and contained only a "Back" hyperlink for returning to the results list. The right subframe displayed the contents of the document (again, with document ID highlighted in red to facilitate cutting and pasting it into the answer window).

The full text of the instructions given to the user is also included in an Appendix.

In order to facilitate user-action collection, the browser's navigation buttons were disabled, but only for Netscape users (due only to lack of knowledge on the experimenter's part of how to do so for Internet Explorer). The five minute time limit per question was not strictly

BEST COPY AVAILABLE

Question → Source	1	2	3	4	5	6	7	8
AP	**200**	**200**	183	112	**200**	**200**	**200**	135
FBIS	100	154	100	100	133	128	100	111
FT	100	145	116	162	133	171	109	100
LA	125	100	183	137	100	100	128	**200**
SJM	175	154	**200**	112	133	128	109	141
WSJ	100	163	**200**	**200**	133	128	152	135

Table 1: Multiplicative Weights for System 2, by Question and Source (largest weights per question in bold)

enforced. Rather, at the five minute mark, a dialog box popped up asking the user to summarize their answer and submit it. For technical reasons, users could actually continue to search after dismissing the dialog box, but in the over 200 searches recorded, this only happened at most 3 times.

3 RESULTS and DISCUSSION

The results and discussion for the experiment is broken down into two parts: first, we will discuss the relative efficacy of the two systems, then we will address the central concern in this study – tracking user actions in an attempt to implicitly identify relevance.

3.1 System 1 vs. System 2

Because System 2 was designed to uprank relevant documents directly (essentially placing documents with the correct answers near the top of the results list for the query formulations used during the first week), our hypothesis is that it will outperform System 1 both in terms of actual answers found by users and also in terms of user satisfaction.

User Performance

We turn our attention first to actual user performance. For comparison purposes, we will use two measures:

1. A binary measure which is 1 for each search session where a user correctly and fully answers the question and provides complete supporting evidence in the form of document IDs, and 0 otherwise.
2. An adjusted measure, which assigns a score as follows:
    - 4 if the user correctly identifies all answers and fully supports them
    - 3 if the user correctly identifies all answers and partially supports them, OR if the user correctly identifies some answers and fully supports them
    - 2 if the user correctly identifies some answers and partially supports them

	Binary		Adjusted	
Question	Sys 1	Sys 2	Sys 1	Sys 2
1	0	.250	.018	.250
2	.071	0	.071	0
3	0	.200	.383	.425
4	0	0	.150	.375
5	.571	.417	.571	.417
6	0	0	0	0
7	.571	.583	.571	.583
8	.133	0	.133	0
Overall	.167	.193	.239	.261

Table 2: Binary and Adjusted Measures per Query and Overall

Table 2 displays these measures on a per-query basis, and also when calculated across all queries. On the binary measure, System 1 does better than System 2 a total of 3 times, and the reverse is also true. Overall, however, System 2 does slightly better. On the adjusted measure, System 2 beats out System 1 on a total of 4 question, whereas the opposite is only true for three questions. Once again, the overall measure favors System 2. So, on both measures, System 2 eeks out a slight lead. These differences still need to be statistically verified.

User Preference

As part of the experiment, users filled out many feedback surveys - one for each question, each system, and an overall survey at the end. In response to the question "Which of the two systems did you like the best overall?" 4 out of 26 subjects liked System 2 better, 1 liked System 1 better, and the rest indicated no preference. When asked "Which of the two systems did you find easier to use?" 3 indicated System 2, 1 indicated System 1, and the rest said no difference. When asked "How different did you find the systems from one another?" 17 found no difference, 2 found little difference, and 6 indicated

BEST COPY AVAILABLE

somewhat of a difference. These results corroborate the performance results from the previous section, indicating that System 2 is only slightly better than System 1.

3.2 Tracking User Actions

On both systems, user's actions were tracked via entries in the HTTP daemon log on the server that implemented the systems. There were three types of salient actions that were recorded:

1. QUERY - when the user pressed the Submit button to issue a query. The user was then shown the results list of document headlines, document IDs, and term counts. The beginning of this list was displayed immediately, and the list continued to grow until all documents were retrieved. This allowed the user to browse the top ranked documents immediately, to facilitate quick searches. This action could be initiated at any time.
2. VIEW - when the user clicked on a document headline in the results list. The frame the results list had been in was replaced with the contents of that document. The user could only perform this action when a results list was displayed.
3. RETURN - when the user clicked the specially generated Back button while viewing a document, thus replacing the document text with the results list from the previous query. The user could only perform this action when a full document was displayed.

Of course, these are only a small subset of all of the user actions that could conceivably be recorded (see for example, [Oard and Kim, 1998]), but represent what was available using the given technology.

Problems

Unfortunately, three design flaws subjected the event logs to a lot of noise when it came time to interpret them. In increasing order of seriousness, they are:

1. Back Button. Although Netscape browsers had the browser back button disabled, Internet Explorer users were free to use it (although they were told not to in the instructions, habits die hard). Thus, the log file could be missing RETURN events. Although it would normally be possible to infer these events, the timing would be unknown, and as it turns out, inferring them isn't possible due to the problems described next.
2. Multiple Processes. Because the user was able to request to VIEW a document before their QUERY had finished processing, it was possible to still have the computer process which was handling the QUERY running at the same time as the process handling the VIEW, which slowed things down a little. This scenario wasn't really a problem, but when the user swapped back and forth between viewing and querying (every RETURN event also initiated a new process to re-generate the results list from scratch), the system could get very bogged down. In and of itself, this would not be a problem for the logging of events, but due to the next problem, it greatly complicated things.
3. End time stamp. Contrary to what one might assume (and what the experimenter did assume), the HTTP daemon does not log the start time of requests, but the finish time. This would not be a problem in a serial, one-process-at-a-time computer. However, when coupled with the previous problem, it nearly rendered the event logs useless.

We have attempted to compensate for the above three flaws in two ways. First, event start times were approximated based on first calculating run-times of processes in standalone mode, and then using a simple algorithm which works backwards from the end of the log, estimating start times based on number of processes running and the standalone runtimes. Second, we combed through the logs and tossed out any of those belonging to any users who exhibited behaviors which resulted in logs that were inconsistent with the constraints above (e.g., ones with two VIEW events in a row or two RETURN events in a row, etc.), on 4 or more of the questions. This left us with the results from 13 users.

3.3 The Myth of Clickthroughs

One of our goals in this experiment was to find a way of inferring which documents were relevant to a query based on the user's actions, without explicitly asking the user for feedback. One common simple assumption is that when a user selects a document for display (an event sometimes called a "clickthrough" in the context of the Web), that document is more likely to be relevant (or, an even simpler version of this heuristic is that the document *is* relevant). In this section, we attempt to debunk that idea. We propose a hypothesis that the amount of time spent reading the document (when adjusted for the document's length) is a more closely correlated with relevance than just whether the document was clicked-through.

In this section, we only analyze the data provided by the 13 "well-behaved" subjects described previously. We are interested in how much time the users spent examining each document that they saw and how that corresponds to relevance. In this section, a document is defined as relevant to a question if the NIST assessors indicated that it supported (at least in part) a correct answer to the question, otherwise it is nonrelevant. As

	Mean	Std Dev
Relevant	0.01808	0.0467
Nonrel	0.00977	0.0159

Table 3: Mean and Standard Deviation of Normalized View Time

a first step in debunking the clickthrough assumption, we note that across all 13 users, a total of 181 VIEW events occurred. Of these, only 66 (or about 36%) were actually for relevant documents. This low number is partially a result of the lack of information the users had to go on: only about 2/3rds of the document's headline was shown. Nevertheless, this argues that there is a very good chance that clicked-through documents are *not* relevant. Perhaps timing information will prove more informative.

Analyzing Normalized View Times

We hypothesize that a user's behavior when reading documents for this task is to scan the document quickly to find an answer or partial answer to the question at hand. They will encounter two kinds of nonrelevant documents, ones which are obviously not relevant and ones which look promising. Relevant documents will also look promising. The amount of time spent on obviously nonrelevant documents will be very small on average. The amount of time spent on promising but nonrelevant documents will likely be a little bit longer than the time spent on relevant documents, because the user will want to scan the entire document to verify the absence of an answer. On the other hand, they will stop scanning a relevant document as soon as they find an answer. Of course, these times will also depend a lot on the length of the document, so we propose normalizing view times by document length.

Thus, we hypothesize that the distribution of normalized view times for nonrelevant documents will be bimodal, with one peak for the obviously nonrelevant and one for the promising but nonrelevant. On the other hand, the distribution of normalized view times for relevant documents will be flatter, and likely have a higher overall mean.

We have calculated the normalized view times by dividing the view time (in seconds) by the number of characters in indexed sections of each document. Table 3 shows the mean and standard deviation for these two distributions. As hypothesized, the average normalized view time for relevant documents is higher than that for nonrelevant. Also, since the distribution of relevant times has a higher variance, it is more spread out, or flatter. The question still remains as to whether the nonrelevant distribution is bimodal or not.

Figure 1 shows two histograms of the nonrelevant normalized view times, which are further normalized by the total number of data points (115) to facilitate comparison to the histograms for relevant document view times. Figure 2 mirrors this for relevant view times. Contrary to our hypothesis, the nonrelevant distribution only looks vaguely bimodal, but in keeping with our hypothesis, the relevant distribution shows signs of normality. Currently, the bins in these histograms have very few (on the order of 10) data points. It is possible that a larger number of data points would better show the true nature of these distributions.

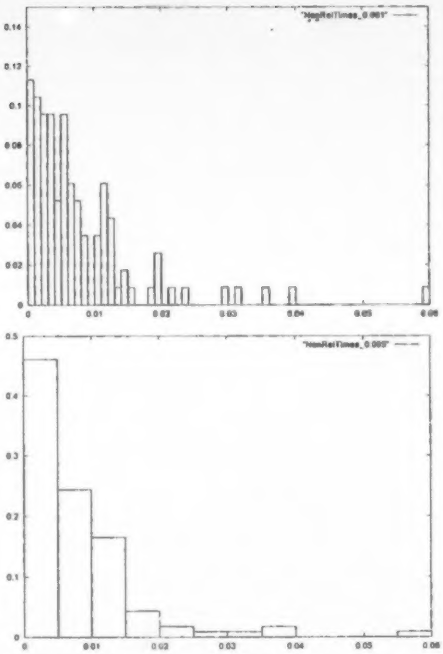


Figure 1: Distribution of Nonrelevant Normalized View Times (bin sizes 0.001 and 0.005)

In the end, what matters is whether or not one can distinguish relevant documents from nonrelevant using normalized view time. Figure 3 shows the receiver operator characteristic curve for using normalized view time as a predictor of relevance. Although the plot shows that it does have some predictive value, the fact that it is not that far from the plot of a random predictor ($y = x$) shows that it is not a great predictor. Clearly, there are other factors that need to be taken into consideration.

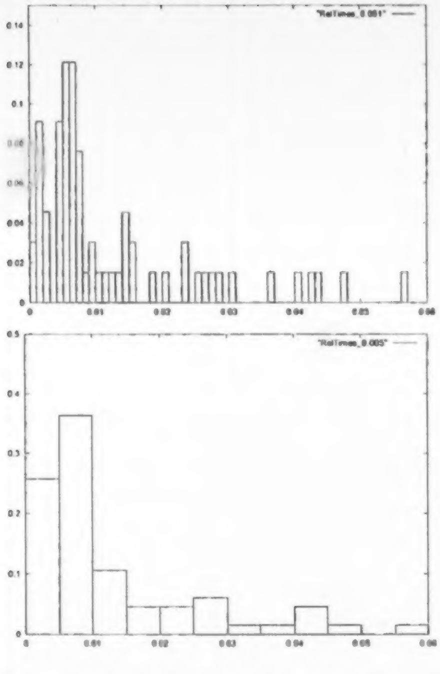


Figure 2: Distribution of Relevant Normalized View Times (bin sizes 0.001 and 0.005)

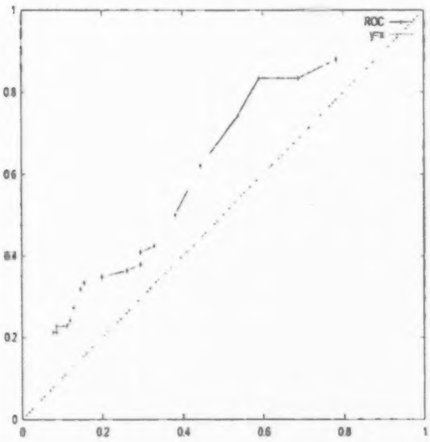


Figure 3: ROC Curve for Using Normalized View Time as Relevance Predictor (False Positive Rate vs. True Positive Rate for Various Theshhold Cutoffs)

4 CONCLUSIONS and FUTURE WORK

If we assume that the steps we took to correct for the errors in the design of our user action logging system were indeed effective, then we have shown that clickthroughs are not necessarily indicative of relevance. In fact, for this experiment, where the preview of a document was quite short (about 2/3rds of a headline), only about one third of the clickthroughs were actually to relevant documents. Clearly, this result depends heavily on our system, the corpus, and the task assigned to the users.

We have introduced the idea of length-normalized viewing time as a predictor of relevance. We have shown that the distribution of this measure differs between relevant documents and nonrelevant documents, although not to such a great extent that it could be used as the single predictor of relevance with real accuracy. Nevertheless, it is an important factor which can more accurately predict relevance than clickthroughs alone.

In the future, we hope to correct the deficiencies of this experiment by creating a better testbed which accurately records event timing, and expands on the types of events logged (e.g., scrolling, mouse-overs, highlighting, etc.), possibly using the Java programming language. It is our belief that a detailed log of user actions could very accurately predict relevance, thus alleviating the need for explicit feedback.

5 Acknowledgments

This research was supported by a Chapman University faculty research and development grant.

References

[Dreilinger and Howe, 1997] Dreilinger, D. and Howe, A. E. (1997). Experiences with selecting search engines using metasearch. *ACM Transactions on Information Systems*, 15(3):195–222.

[Morita and Shinoda, 1994] Morita, M. and Shinoda, Y. (1994). Information filtering based on user behavior analysis and best match text retrieval. In Croft, W. B. and van Rijsbergen, C., editors, *SIGIR 94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 272–281, Dublin. Springer-Verlag.

[Oard and Kim, 1998] Oard, D. W. and Kim, J. (1998). Implicit feedback for recommender systems. In *Proceedings of the AAAI Workshop on Recommender Systems*, Madison, WI.

[Salton, 1971] Salton, G., editor (1971). *The SMART Retrieval System – Experiments in Automatic Document Retrieval.* Prentice-Hall Inc., Englewood Cliffs, N.J.

6 Appendix

On the next few pages are the instructions given to subjects, including an ASCII rendition of the Web interface.

Instructions

Please read these instructions carefully. Please reread them multiple times until you feel comfortable with what they say. You will not see these instructions again during the course of the experiment. If you want, you can print this page out and/or copy-and-paste it to another window for reference when you're searching.

Overview of the Experiment

The goal of this experiment is to determine how well an information retrieval system can help you to answer questions you might ask when searching newswire or newspaper data. The questions are of two types:

- Find a given number of different answers. For example: Name 3 hydroelectric projects proposed or under construction in the People's Republic of China.
- Choose between two given answers. For example: Which institution granted more MBAs in 1989 - the Harvard Business School or MIT-Sloan?

You will be asked to search on four questions with one system and four questions with another. These two sessions might be separated by a period of up to a week. You will have **five minutes** to search on each question, so *plan your search wisely* (you will be shown the question before you have to start searching). You will be asked to answer the question and provide a measure of your certainty of your answer both before and after searching, and to indicate which documents were helpful for determining the answer.

You will also be asked to complete several additional questionnaires:

- Before the experiment - computer/searching experience and attitudes
- After each question
- After each four questions with the same system
- After the experiment - system comparison and experiment feedback

System Details

The searching system you will be using is similar to the typical Web search engine. You will type keywords into a text box, press a "Search" button, and then view a list of the titles of documents ranked according to how well the document matches your search query.

The screen will be divided into two portions. In the top portion, you will see the following elements: the question you are looking to answer, a search textbox and button, an area for typing in your answer to the question, and a drop-down list used to indicate how certain you are of your answer. There is also a countdown timer that indicates how long you have to complete your search (this is also displayed in the status bar at the bottom of your browser). The bottom half of the screen serves a dual purpose:

- It displays the list of results of a search, which consists of a sorted list of document IDs and titles, along with how many times each word in your query appears in that document.
- It also displays the full content of individual documents whenever you click on their title in the results list.

An example of what the screen would look like at some point in a search is shown on the last page of these instructions.

Every document in the database has a unique identifier, typically 2-4 letters which indicate the source (e.g., WSJ for Wall Street Journal), followed by a possibly hyphenated series of numbers. These will be displayed in a red font for easy identification. For example, WSJ911231-0122. It is **very important** that you include the document identifiers for those documents in which you found supporting evidence as part of your answer. Otherwise, we won't be able to verify your answer.

Verification of your answer will be done by human reviewers. Unfortunately, you will receive no immediate feedback as to whether your answer was correct, but we will let you know how you did after the manual scoring is done (some time this fall).

Hints and Special Requests

- **Please do not use the navigation buttons (Back, Forward, Reload, Refresh, etc.) on your browser.** Click only on buttons and links that are in the main portion of your browser screen.
- **Please observe the 5 minute time limit** - when the system tells you to submit your answer, do so without delay. This is a scientific experiment, so certain guidelines need to be followed by all subjects to ensure the results are accurate. The time you spend searching is logged, and if you spend significantly more than 5 minutes, we can't use your data.
- Some users of this system have noted the following ideas for fast and efficient use of the system:
  - Maximize the size of your browser window (you can do that right now if you want).
  - When you get through the initial questionnaire and are on the search screen, scroll the top frame of the browser window so you can see both the search box and the answer box.

- After issuing a query, the number of candidate documents might be quite large - you may have to do some scrolling to find what you're looking for. To help you, a portion of the document's headline will be displayed, along with how many times each word in your query appears in that document. This should help you quickly eliminate irrelevant documents.
- You don't have to wait for the whole results list to be displayed before choosing a document to examine - you can click on the document headline at any time.
- Use the browser's "Find" feature to quickly locate relevant parts of a document that is currently being displayed in the lower half of the screen (the keyboard shortcut for Find is usually Ctrl-F or Command-F)
- If one word is more important than others in your query, repeat it multiple times (e.g. "dog dog dog bones").
- It is often useful to modify your query by removing bad keywords (i.e., ones that cause the system to retrieve irrelevant documents) and adding new keywords. When you're reading a document, be on the lookout for specific terms that you may want to add to your query. If you're getting bad results, don't be afraid to modify and re-issue a query.
- This search engine does NOT support any of the fancy searching features you may be used to (e.g., AND, OR, +, enclosing phrases in quotes, etc.). It only allows you to type in keywords
- Use your browser's copy-and-paste facilities to copy document identifiers to the answer box (usually, Ctrl-C and Ctrl-V)

You should have been mailed a four-character username via email. Make sure you have about an hour's worth of time available and a consistent and good internet connection, and then enter your username to begin:

If you are having problems, please contact Chris Vogt

Example screen (rendered in ASCII) begins on next page:

```
=====================================================================================
3:14 Minutes Remaining
Please answer the following question by searching for relevant documents:
Which institution granted more MBAs in 1989 - the Harvard Business School or MIT-Sloan?
+-------------------------+
| harvard sloan mba       |  {Search}
+-------------------------+
Enter your answer in the space below along with one or more documents
that supports the answer.  (You must list at least one document that
answers the question.)
+-----------------------------------------------------------+
| MIT-Sloan FT944-15805 WSJ881104-0152                      |
|                                                           |
|                                                           |
+-----------------------------------------------------------+
Please choose how certain you are about your answer:
o Extremely Uncertain
o Somewhat Uncertain
o Neutral
o Somewhat Certain
o Extremely Certain
{Submit Answer}
=====================================================================================
DocID            mba  harvard  sloan  Title
FT944-15805      124  3        .   .  -Survey of Business Schools - An A-Z Guide (
AP881017-0012    .    32       .      -Presidential Campaign Elevates Elite Rival
FT924-176        22   1        .      -Management: Successful to a degree - Critics
FT932-12256      18   1        .      -Survey of Business Schools (2): Some employ
LA093090-0082    .    .        16     -ARTS FESTIVALS; IT'S SAN FRANCISCO'S TURN
LA113089-0061    .    14       6      -HARVARD-WESTLAKE MERGER IS FOCUS OF LAWSU
FT941-2127       12   6        .      -Survey of Management Education &amp; Trainin
WSJ881104-0152   .    1        14     -MIT's Sloan School Tries To Drop a Name
WSJ911007-0114   .    22       .      -Harvard Marks Down Speculative Fund by $
LA111989-0153    .    21       .      -COLLEGE FOOTBALL; THE SCENE AT ANOTHER RI
LA110289-0081    .    21       .      -PARENTS' REVOLT DELAYS MERGER OF 2 SCHOOL
LA113089-0094    .    12       6      -BATTLE GROWS OVER SCHOOLS' MERGER; EDUCAT
FT941-2136       16   .        .      -Survey of Management Education &amp; Trainin
FT932-12258      13   2        .      -Survey of Business Schools (1): Horses for
WSJ900827-0138   10   3        .      -Manager's Journal: How Much Is an MBA Re
FT944-14774      13   1        .      -Management: Degree of reluctance - George B
WSJ870615-0133   10   2        .      -Manager's Journal: The Value of Today's
FT922-13885      12   1        .      -Survey of Management Education and Training
FT944-9861       .    .        10     -Management: Pioneers and prophets - Alfred P
=====================================================================================
```

BEST COPY AVAILABLE

TREC-9 CLIR at CUHK
Disambiguation by Similarity Values Between Adjacent Words

Honglan Jin   Kam-Fai Wong

Systems Engineering and Engineering Management Department
The Chinese University of Hong Kong
{hljin, kfwong@se.cuhk.edu.hk}

Abstract

We investigated the dictionary-based query translation method combining the translation disambiguation process using statistic co-occurrence information trained from the provided corpus.

We believe that neighboring words tend to be related in contextual meaning and have higher chance of co-occurrence particularly if adjacent words (two or more) compose a phrase. The correct translation equivalents of co-occurrence pattern in a source language are more likely to co-occur in a target language documents than in conjunction with any incorrect translation equivalents within a certain range of contextual window size.

In this work, we tested several methods to calculate the degree of co-occurrence and used them as the basis of disambiguation. Different from most disambiguation methods which usually select one best translation equivalent for a word, we select the best translation equivalent pairs for two adjacent words. The final translated queries are the concatenation of all overlapped adjacent word translation pairs after disambiguation.

System Description

The well-known vector space modeled SMART information retrieval system, Version 11, is used as our platform. We adopted the weighting strategy for documents and queries as *Lnu.Ltu* [1,2], which has been proved more successful than cosine normalization.

The queries were produced after query translation and ambiguity resolution. We fed them to the SMART system to get the retrieval result.

Query Translation

Bilingual Dictionaries

A bilingual English to Chinese machine readable dictionary (MRD) produced by Earth Village (http://www.samlight.com/ev/eng/) is used as our translation resource. This MRD has many entries exactly the same with those in the bilingual dictionary edited by LDC (http://morph.ldc.upenn.edu/Projects/Chinese) The reason we chose Earth Village was that Earth Village provided POS (parts of speech) information. We thought it was useful at the early stage of our experiments but ended up not using it. For phrase translation purpose, we combined three sources: a Chinese to English MRD (http://www.mindspring.com/~paul_denisowski/cedict.html), Earth Village, and the one from LDC. The Chinese to English MRD was converted to English->Chinese and we extracted all the phrase translations from the above resources and complied a single phrase level English Chinese MRD. From our previous experiments, we found the more the number of translations for a word, the higher the chance of introducing extraneous translations for this word. For this reason, we used only Earth Village MRD as our word level translation resource. There are 61729 entries, 2.3 translations in average for each word. However, for the 25 queries in TREC-9, each word in the source language (English) has 5 translations in Chinese after the translation by Earth Village MRD.

Phrase level translation was performed before word level translation. All English words were morphologically transformed to its original word root by using WordNet (http://www.wordnet.com). The root was used as the key to search for its corresponding translations in the dictionary. To perform phrase level translation, we created from bigrams to five-grams composed by adjacent words first in the queries. If a higher gram translation failed, a lower one would be tried until bigram was reached. If it still failed, word level translation was adopted. Otherwise, phrase level translation was performed a nd the same procedure starting from the next word position was repeated.

Chinese Segmentation

The corpus and the translated queries were segmented by using the perl coded software developed by Erik Peterson (http://www.mandarintools.com). But we replaced the original word list dictionary with our own, a word list of Hong Kong style words.

Post processing after translation

After the initial translation, we did some pruning based on our previous experience and some ad hoc rules. Earth Village is basically a mainland Chinese language style dictionary while the corpus used is in Hong Kong style Chinese. For the same concept, two styles may have totally different representations in the bilingual dictionary. For the translated segmented queries (mainland style), we did the following pruning:

1. Delete the translations having longer than five Chinese characters unless there 's only one translation: If a translation is too long (exceed five characters for example), this translation is highly likely the description of the word meaning instead of direct translation of the word.
2. Delete the translated entries being segmented unless there 's only one translation. If a translated word is segmented, very probably it is because (1) there's no entry in the dictionary for the word segmentation, (2) it has different translations in China and Hong Kong.
3. Keep only the first three translations with the highest term frequency (*TF*) in the corpus. From our previous experiments, translations with high term frequency in the target language tend to have higher chance of being the correct translations than rare appearing ones.

After the above processing, each word has no more than three translation candidates.

Disambiguation

There are several scenarios of resolving translation ambiguity by using co-occurrence (CO) information.

First, a NLP parser can be used to recognize all the grammatical sub-components such as a phrase. Then the CO information is used to calculate the coherent values in the target language among the composite words within a phrase. The translation for this phrase is the one that has the highest coherent values among all the translation combinations for the phrase. However, a parser is not always reliable. Further, individual words which are not associated to any phrases are isolated in meaning ; we can do nothing to resolve their translation ambiguity.

Second, ambiguity is resolved in sentence level rather than phrase level like method one. We create all the translation combinations in the target language for a sentence and choose the one that has the highest coherent values as the final translation. Obviously, as a sentence is usually much longer than a phrase, the number of translation combinations in this method is much larger and thus the computation cost can be too high. Another problem with this method is that when the corpus is not large enough, the coherent values trained from it may be misleading. The longer a sentence is, the more costly is the computation and the larger the corpus is required . The rate of increase of both computation cost and size of the corpus required is exponential.

Third, the disambiguation is done between two adjacent words. Among all the translation combinations between two words, we choose the pair with the highest coherent values as the final translation. The cost is low and the corpus size requirement is much less restricted. We adopted this method for its easy computation and the corpus condition.

Co-occurrence information such as mutual information (MI) [3] was used to calculate the degree of cohesion between two words. MI measure however strongly favors rarely appearing words. We apply the methods to calculate the similarity values between all adjacent word pairs in queries to reduce the translation errors.

If two words always co-occur within a particular contextual range such as adjacent positions, a sentence or even a whole document, they should have similar distribution pattern within that contextual range throughout the document collection. Higher similar distribution means higher degree of co-occurrence pattern or coherent values. The correct translation equivalents of co-occurrence pattern in source language is more likely to co-occur in the target language documents than in conjunction with any incorrect translation equivalents within a certain range of contextual window size.

We calculate this degree of similarity as the inner product of two vectors each representing a word distribution in the collection. For disambiguation purpose, a fine-grained context for a co-occurrence scope is essential. We chose the window size to be a sentence in target language. The dimension of the vectors are the number of windows (or the number of sentences in the collection). The value of each dimension is 1 if a word appears in that sentence and 0 otherwise. We made two assumptions here: a word always appears no more than once in a sentence and the variation of sentence length can be ignored. By considering only the distribution throughout the corpus as the normalization factor, we assigns *idf* value to each dimension of a vector of a word as the weight, i.e.,

$$idf = \log(N / N_c)$$

where N is the total number of documents in the corpus and Nc is the number of documents where the word appears. The similarity of two words by their inner product is the sum of

$$tf(ab) * idf(a) * idf(b)$$

in each dimension where $tf(ab)$ is the co-occurrence indicator (1 or 0) in sentence scope and $idf(a)$, $idf(b)$ are the idf values for words a and b respectively.

We calculate similarity values for all possible pairs of translation between two adjacent non-stopword words in queries and select the translated pairs with the highest similarity value in the target language as the final translations.

The final translated queries are the concatenation of all overlapped adjacent word translation pairs after disambiguation.

Our method is different from others in that we did not select the best translation candidate for a word. We select the best translation pairs instead. By considering all overlapping pairs, each word in fact has two translations (except the first and the last words in a sentence). But if a translation has strong similarity value with the translation of the word adjacently before and after it, two translations should be the same.

There are several features for this arrangement: First, no grammatical boundary such as phrase boundary recognition is needed during disambiguation. Second, even if two adjacent words are not a phrase, many of them are related in contextual meaning and have a higher chance of co-occurrence. Overlapped concatenation makes each word's translation be selected twice. If two translations are the same, such a word appearing in queries in the target language would have higher weighting than a word having two different translations when the *TF* value is considered in query weighting. We believe the former case would produce more correct translations. If this is the case, more correct translations would enforce higher weighting values, which would help the retrieval performance.

Experiments

We submitted two official runs. One was monolingual and the other cross-lingual. However, we will describe more runs here to support our analysis.

We used all three parts of a query: title, description and narratives. All our queries are long queries. We used SMART *Lnu.Ltu* weighting and SMART Rocchio query expansion (mono run) before and after query translation (xlingual run). Three parameters of expansion were set to alpha=8, beta=16, gamma=8. For monolingual query expansion, we added 35 terms extracted from the top 10 documents. From our previous experiment, we trained the optimal number of terms to be 10 terms. But as there was a copyright statement at the end of each document, we increased the number to 35 terms. For the same reason, the number to be added for cross-lingual run is increased from 20 terms to 50 terms from the top 20 documents. We also did query expansion before query translation using the corpus from TREC data vol. 5, the Foreign Broadcast Information Service (FBIS) files. FBIS is more than 400 MB in size and contains many international related documents. The number of expanded terms were 10 terms from the top 10 documents. The translation for the added terms in the source language were done by selecting the first two translations in the dictionary. All the parameters mentioned above were trained from our previous experiments if not otherwise stated.

Table 1:Official run results

Run	11-point	Relevant	R precision
CHUHK00CH1	0.2419	552	0.2524
CHUHK00XEC1	0.2583	514	0.2618

Table 1 is our official run results. **CHUHK00CH1** is the **monolingual** run and **CHUHK00XEC1** is the **cross-lingual run**. There are 663 relevant documents altogether in TREC-9. Table 2 is the component result for monolingual CHUHK00CH1 run and Table 3 is the component result for cross-lingual CHUHK00XEC1 run.

Table 2: Monolingual component results

Run	11-point
Lnu.Ltu	0.2288
above+expansion (top 10 docs, 35 terms)	0.2419 (+6%)

Table 3: Cross-lingual component results

Run	11-point
Lnu.Ltu	0.1862
above+expansion before query translation (top 10 docs, 10 terms)	0.2642
above+expansion after query translation (top 20 docs, 50 terms)	0.2583

There are some interesting phenomena from the results. Our final cross-lingual run exceeds its corresponding monolingual run, the performance ratio is 0.2583/0.2419=106.8%. However, if we compare the performance before any query expansions, that ratio is 0.1862/0.2288=81%.

For the cross-lingual run, the improvement of query expansion (from 0.1862 to 0.2642) before query translation is as high as 42%. We contribute the drastic improvement to the following reasons: First, the corpus "Foreign Broadcast Information Service" seems to contain many relevant documents to the queries in the source language and thus it is ideal for the source of blind relevance feedback. Second, selecting the first two translations for the expanded terms seems to be very successful in this context. Due to the time limitation, we could not investigate carefully on how to select the best translation candidates for isolated terms.

By looking at the results produced from the final query expansion, the improvement for monolingual is 6% (from 0.2288 to 0.2419), which is reasonable. However, the query expansion after translation led to performance degradation, from 0.2642 to 0.2583 even though the retrieved relevant documents increased from 495 to 514.

Table 4 concludes our performance comparing with other groups.

Table 4: Result comparison

Run	best	median	worst
CHUHK00CH1(mono)	3	12	10
CHUHK00XEC1(xlingual)	2	16	7

Analysis

In this section, we present the results from more runs to support our analysis. We aim to compare our proposed method with other related ones such as MI (mutual information) and highest term frequency methods. To do this, we did the following experiments.

1. The disambiguation is done by selecting the translation pairs with the highest MI value (denoted as **sim_mi**). MI is calculated as

$$I(a,b) = \log_2 \frac{P(x,y)}{p(x,y)}$$

2. The disambiguation is done by selecting the translation candidate with the highest term frequency appeared in the target corpus (denoted as **htf**). The similarity measure used in our official runs was used here except *idf* normalization, i.e., the disambiguation is done by selecting the translation pairs with the highest value of co- occurrence numbers (denoted as **sim_tf**).

Table 5 shows the retrieval results for the above runs in average precision (11-point). These runs were all done by query expansion before and after query translation with the same parameters used in our official cross-lingual runs.

Table 5: Comparative results

Run	11-point (b)	11-point (a)
mi	0.2552	0.2473
htf	0.2613	0.2544
sim_tf	0.2638	0.2564

MI method was worse than the others while htf, sim_tf and our sim_idf performed better. It is surprising that htf, the simplest method produced such a good result considering the efforts it takes. The result of sim_tf reveals similar message: high term frequency translations in the target are good indication of good translations. MI has the disadvantage of strongly favoring rarely appearing words.

We performed a final experiment trying to support our hypothesis that our overlapped concatenation of best selected translation pairs would enforce more correct translations to have higher weighting if the term frequency factor in the query is properly considered. If this is the case, it would be helpful for retrieval performance. To test this, we did Lnu.ntu weighting retrieval where term frequency factor is "augmented" comparing with Lnu.Ltu weighting.

The average 11-point recall precision is **0.2649** before the query expansion and **0.2596** after the query expansion. Although the increase is not obvious (0.2642 and 0.2583 in our official cross-lingual run), this result gives the highest figure comparing with all Lnu.Ltu runs.

We also observed consistent retrieval degradation after the final query expansion in all cross-lingual runs.

Conclusion:

We presented our disambiguation method by using similarity values between all adjacent words in the target language. It is based on the co-occurrence numbers within a sentence scope in the whole collection. On top of that, *idf* values of a word pair are used to normalize the co-occurrence numbers. We have shown that both co-occurrence number with or without normalization worked better than MI method. In particular, *idf* normalization is 4.5% (0.2583/0.2473) better than MI method in our experiments. More tests will be performed to further verify the improvement reported here.

This is our first participation in TREC. We reckon that this is a good start for our future research.

Acknowledgements

This project is partially supported by DARPA, USA, under the TIDES program (grant No.: N6601-00-1-8912) and Research Grant Committee, Hong Kong, under the direct grant initiative (project No.: 2050253).

References

[1] Amit Singhal, Chris Buckley, Mandar Mitra. Pivoted Document Length Normalization. In Proceedings of the 19 th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 21-29, 1996.

[2] Chris Buckley, Amit Singhal, Mandar Mitra, Gerard Salton. New Retrieval Approaches Using SMART: TREC 4. In D.K. Harman, editor, *Proceedings of the Fourth Text REtrieval Conference (TREC-4).* NIST Special Publication 500-236, 1996.

[3] Church, K.W. and Hanks, P. 1990. 'Word collocation norms, mutual information, and lexicography' *Computational linguistics* 16: 22-29

Syntactic Clues and Lexical Resources in Question-Answering

Kenneth C. Litkowski
CL Research
9208 Gue Road
Damascus, MD 20872
ken@clres.com

Abstract

CL Research's question-answering system (DIMAP-QA) for TREC-9 significantly extends its semantic relation triple (logical form) technology in which documents are fully parsed and databases built around discourse entities. This extension further exploits parsing output, most notably appositives and relative clauses, which are quite useful for question-answering. Further, DIMAP-QA integrated machine-readable lexical resources: a full-sized dictionary and a thesaurus with entries linked to specific dictionary definitions. The dictionary's 270,000 definitions were fully parsed and semantic relations extracted to provide a MindNet-like semantic network; the thesaurus was reorganized into a WordNet file structure. DIMAP-QA uses these lexical resources, along with other methods, to support a just-in-time design that eliminates preprocessing for named-entity extraction, statistical subcategorization patterning, anaphora resolution, ontology development, and unguided query expansion. (All of these techniques are implicit in DIMAP-QA.)

The best official scores for TREC-9 are 0.296 for sentences and 0.135 for short answers, based on processing 20 of the top 50 documents provided by NIST, 0.054 and 0.083 below the TREC-9 averages. The initial post-hoc analysis suggests a more accurate assessment of DIMAP-QA's performance in identifying answers is 0.485 and 0.196. This analysis also suggests that many failures can be dealt with relatively straightforwardly, as was done in improving performance for TREC-8 answers to 0.803 and 0.597 for sentences and short answers, respectively.

1. Introduction

TREC-9 DIMAP-QA proceeded from last year's version by first removing many shortcomings noted there (where it was suggested that the official 250-byte, or sentence, score of 0.281 could be raised to an estimated 0.482) by including documents not processed, resolving parsing problems affecting both questions and documents, and resolving triple extraction problems. Dealing with these problems improved the score to 0.550. DIMAP-QA was then extended to extract 50-byte answers, with feedback to the sentence extraction (i.e., when a viable short answer was recognized, its sentence was given a higher score). This extension focused on developing question-specific routines for extracting short answers based on the discourse entities and the types of semantic relations in which they participated. This improved scores to 0.740 for sentences and 0.493 for short answers, suggesting that a substantial portion of question-answering can be achieved without special pre-processing. At this point in development, the lexical resources were integrated. Although these resources could have been used directly to answer questions, the just-in-time model used them instead for substantiation. For example, in "where" questions, definitions provided a background set of discourse entities used in evaluating document sentences. For "what" questions (e.g., "what country"), dictionary definitions were examined to determine whether a document discourse entity was defined as or had the hypernym "country". If no match, the thesaurus was examined to determine if the hypernym for a document discourse entity was in the same thesaurus category (e.g., as "country" where "Belgium" is defined as a "kingdom"). Incorporation of these lexical resources improved the TREC-8 scores to 0.803 for sentences and 0.597 for short answers.

DIMAP-QA is a part of the DIMAP dictionary creation and maintenance software, which is primarily designed for making machine-readable dictionaries machine-tractable and suitable for NLP tasks, with some components intended for use as a lexicographer's workstation.[1] The TREC-9 QA track provided an opportunity for experimenting with the limits of question-answering based only on syntactical clues and for examining use of computational lexical resources (dictionary and thesaurus). The development of the system for TREC-9 and the analysis of failures provides a good delineation of the limits of different types of evidence and the role of lexical resources.

---

[1] DIMAP, including the question-answering component, is available from CL Research. Demonstration and experimental versions are available at http://www.clres.com.

2. Problem Description

Participants in the TREC-9 QA track were provided with 693 unseen questions to be answered from the TREC CD-ROMs, (about 1 gigabyte of compressed data), containing documents from the *Foreign Broadcast Information Service*, *Los Angeles Times*, *Financial Times*, *Wall Street Journal*, *Associated Press Newswire*, and *San Jose Mercury News*. These documents were stored with SGML formatting tags. Participants were given the option of using their own search engine or of using the results of a "generic" search engine. CL Research chose the latter, relying on the top 50 documents retrieved by the search engine. These top documents were provided simultaneously with the questions.

Participants were then required to answer the 693 questions in either 50-byte answers or by providing a sentence or 250-byte string in which the answer was embedded. For each question, participants were to provide 5 answers, with a score attached to each for use in evaluating ties.[2] NIST evaluators then judged whether each answer contained a correct answer. Scores were assigned as the inverse rank. If question q contained a correct answer in rank r, the score received for that answer was 1/r. If none of the 5 submissions contained a correct answer, the score received was 0. The final score was then computed as the average score over the entire set of questions.

CL Research submitted 4 runs, 2 each for the 250- and 50-byte restrictions, one analyzing only the top 10 documents and the other only the top 20 documents, to examine whether performance was degraded in going from 10 to 20 documents.

3. System Description

The CL Research prototype system consists of four major components: (1) a sentence splitter that separated the source documents into individual sentences; (2) a parser which took each sentence and parsed it, resulting in a parse tree containing the constituents of the sentence; (3) a parse tree analyzer that identified important elements of the sentence and created semantic relation triples stored in a database; and (4) a question-answering program that (a) parsed the question into the same structure for the documents, except with an unbound variable, and (b) matched the question database records with the document database to answer the question. The matching process first identified candidate sentences from the database, extracted short answers from each sentence, developed a score for each sentence, and chose the top 5 sentences (and their short answers) for submission.

3.1 Sentence Identification in Documents

The parser (described more fully in the next section) contains a function to recognize sentence breaks. However, the source documents do not contain crisply drawn paragraphs that could be submitted to this function. Thus, a sentence could be split across several lines in the source document, perhaps with intervening blank lines and SGML formatting codes. As a result, it was first necessary to reconstruct the sentences, interleaving the parser sentence recognizer.

At this stage, we also extracted the document identifier and the document date. Other SGML-tagged fields were not used. The question number, document number, and sentence number provided the unique identifier when questions were answered.

TREC-9 added 3 document collections (*Wall Street Journal*, *Associated Press Newswire*, and *San Jose Mercury News*). Although we had tested processing of these document types before the test suite was made available, we had not captured nuances not described in the DTDs. As a result, there were many "bombs" that occurred in processing the top documents; many of the problems had to be fixed during the final processing. Although this violates the strict rule against making changes after the questions are made available, these changes did not go to the heart of the question-answering, but only to the ability of the system to process the documents. After submission, further nuances affecting system performance were identified, most notably in the omission of important textual material ("lead paragraphs") in the *Wall Street Journal* and the *San Jose Mercury News* and the combining of multiple sentences from *Associated Press* documents (because of the way quoted material was handled). These problems had an effect on performance, as described below.

---

[2] Although this statement appears in one of the problem specifications, the score is not used and only the position of the answer is considered.

For the TREC-9 QA runs submitted to NIST, the top 20 documents (as ranked by the search engine) were analyzed (30 were processed for 250 of the questions). Overall, this resulted in processing 14605 documents (up from 1977 in TREC-8) from which 422,562 (up from 63,118) sentences were identified and presented to the parser. Thus, we used an average of 28.9 (down from 31.9) sentences per document or 290 sentences for the 10-document set, 580 for the 20-document set, and 870 for the 30-document set for each question.

3.2 Parser

The parser in DIMAP (provided by Proximity Technology, Inc.) is a grammar checker that uses a context-sensitive, augmented transition network grammar of 350 rules, each consisting of a start state, a condition to be satisfied (either a non-terminal or a lexical category), and an end state. Satisfying a condition may result in an annotation (such as number and case) being added to the growing parse tree. Nodes (and possibly further annotations, such as potential attachment points for prepositional phrases) are added to the parse tree when reaching some end states. The parser is accompanied by an extensible dictionary containing the parts of speech (and frequently other information) associated with each lexical entry. The dictionary information allows for the recognition of phrases (as single entities) and uses 36 different verb government patterns to create dynamic parsing goals and to recognize particles and idioms associated with the verbs (the context-sensitive portion of the parser).

The parser output consists of bracketed parse trees, with leaf nodes describing the part of speech and lexical entry for each sentence word. Annotations, such as number and tense information, may be included at any node. The parser does not always produce a correct parse, but is very robust since the parse tree is constructed bottom-up from the leaf nodes, making it possible to examine the local context of a word even when the parse is incorrect. In TREC-9, parsing exceptions occurred for only 69 sentences out of 422562 (0.0002, down from 0.008), with another 131 "sentences" (usually tabular data) not submitted to the parser. Usable output was available despite the fact that there was at least one word unknown to the parsing dictionary in 33,467 sentences (7.9 percent).

3.3 Document and Question Database Development

A key step of DIMAP-QA is analysis of the parse tree to extract semantic relation triples and populate the databases used to answer the question. A **semantic relation triple** consists of a discourse entity, a semantic relation which characterizes the entity's role in the sentence, and a governing word to which the entity stands in the semantic relation. A triple is generally equivalent to a logical form (where the operator is the semantic relation) or a conceptual graph, except that a semantic relation is not strictly required, with the driving force being the discourse entity.

The first step of discourse processing is identification of suitable discourse entities. For TREC-8, this involved analyzing the parse tree **node** to extract numbers, adjective sequences, possessives, leading noun sequences, ordinals, time phrases, predicative adjective phrases, conjuncts, and noun constituents as discourse entities. To a large extent, named entities, as traditionally viewed in information extraction, are identified as discourse entities (although not specifically identified as such in the databases). For TREC-9, the parse output was further mined, more fully exploiting the syntactic relations between sentence constituents. The most notable of these was the characterization of various forms of appositives (parenthesized expressions, relative clauses, and true appositives), which frequently provide the answers to questions.

The semantic relations in which entities participate are intended to capture the semantic roles of the entities, as generally understood in linguistics. This includes such roles as agent, theme, location, manner, modifier, purpose, and time. For TREC-9, we did not fully characterize the entities in these terms, but generally used surrogate place holders. These included "SUBJ," "OBJ", "TIME," "NUM," "ADJMOD," and the prepositions heading prepositional phrases. Appositive phrases were characterized by identifying the sentence word they modified and the beginning and ending words of the phrase; their use is described particularly for answering Who and What questions.

The governing word was generally the word in the sentence that the discourse entity stood in relation to. For "SUBJ," "OBJ," and "TIME," this was generally the main verb of the sentence. For prepositions, the governing word was generally the noun or verb that the prepositional phrase modified. (Because of the context-sensitive dynamic parsing goals that were added when a verb or a governing noun was recognized, it was possible to identify what was modified.) For the adjectives and numbers, the governing word was generally the noun that was modified.

The semantic relation and the governing word were not identified for all discourse entities, but a record for each entity was still added to the database for the sentence. Overall, 4,149,106 semantic relation triples were created (up from 467,889) in parsing the 422,562 sentences, an average of 9.8 triples per sentence (up from 7.4 in TREC-8).

The same functionality was used to create database records for the 693 questions. The same parse tree analysis was performed to create a set of records for each question. The only difference is that one semantic relation triple for the question contained an unbound variable as a discourse entity, corresponding to the type of question. The question database contained 2272 triples (for 693 questions), an average of 3.3 triples per question. This is down from 4.5 triples per question in TREC-8. This is indicative of the fact that the questions were "simpler", making them more difficult to answer, since there was less information on which to match.

3.4 Lexical Resources

A major addition to the question-answering system for TREC-9 QA was the integration of a machine-tractable dictionary and thesaurus. These were provided in machine-readable form by The Macquarie Library Pty Ltd of Australia. The dictionary, known as Big Mac, was converted into a format suitable for uploading into DIMAP dictionaries, during which most of the raw data were put into specific fields of a DIMAP dictionary (e.g., headword, part of speech, definitions, example usages, and many "features" characterizing syntactic properties and other information, particularly a link to Macquarie's thesaurus and identification of a "derivational" link for undefined words to their root form).

After conversion and upload, the entire dictionary of 270,000 definitions was parsed to populate the raw dictionary data by adding semantic relations links with other words. The most important result was the identification of the hypernyms of each sense. Other relations include synonyms (discernible in the definitions), typical subjects and objects for verbs, and various semantic components (such as manner, purpose, location, class membership, and class inclusion). This dictionary, accessed during the question-answering process, is thus similar in structure to MindNet (Richardson, 1997).

The Macquarie thesaurus was provided in the form of a list of the words belonging to 812 categories, which are broken down into paragraphs (3 or 4 for each part of speech) and subparagraphs, each containing about 10 words that are generally synonymous. We were also provided (Green, 2000) with a set of perl scripts for inverting the thesaurus data into alphabetical order, where each word or phrase was listed along with the number of entries for each part of speech, and an entry for each distinct sense identifying the category, paragraph, and subparagraph to which the word or phrase belongs.

The resultant thesaurus is thus in the precise format of the combined WordNet index and data files ((Fellbaum, 1998)), facilitating thesaurus lookup.

3.5 Question Answering Routines

For TREC-9, a database of documents was created for each question, as provided by the NIST generic search engine. A single database was created for the questions themselves. The question-answering consisted of matching the database records for an individual question against the database of documents for that question.

The question-answering phase consists of three main steps: (1) coarse filtering of the records in the database to select potential sentences, (2) detailed analysis of the question to set the stage for detailed analysis of the sentences according to the type of question, establishing an initial score of 1000 for each sentence, (3) extracting possible short answers from the sentences, with some adjustments to the score, based on matches between the question and sentence database records and the short answers that have been extracted and (4) making a final evaluation of the match between the question's key elements and the short answers to arrive at a final score for the sentence. The sentences and short answers were then ordered by decreasing score for creation of the answer files submitted to NIST.

3.5.1 Coarse Filtering of Sentences

The first step in the question-answering phase was the development of an initial set of sentences. The discourse entities in the question records were used to filter the records in the document database. Since a discourse entity in a record could be a multiword unit (MWU), the initial filtering used all the individual words in the MWU. Question and sentence discourse entities were reduced to their root form, eliminating issues of tense and number. All words were reduced to lowercase, so that issues of case did not come into play during this filtering step. Finally, it was not necessary for the discourse entity in the sentence database to have a whole word matching a string from the question database. Thus, in this step, all records were selected from the document database having a discourse entity that contained a substring that was a word in the question discourse entities.

MWUs were analyzed in some detail to determine their type and to separate them into meaningful named entities. We examined the capitalization pattern of a phrase and whether particular subphrases were present in the Macquarie dictionary. We identified phrases such as "Charles Lindbergh" as a person (and hence possibly referred to as "Lindbergh"), "President McKinley" as a person with a title (since "president" is an uncapitalized word in the Macquarie dictionary), "Triangle Shirtwaist fire" as a proper noun followed by a common noun (hence looking for either "Triangle Shirtwaist" or "fire" as discourse entities).

The join between the question and document databases produced an initial set of unique (document number, sentence number) pairs that were passed to the next step.

3.5.2 Identification of Key Question Elements

As indicated above, one record associated with each question contained an unbound variable as a discourse entity. The type of variable was identified when the question was parsed and this variable was used to determine which type of processing was to be performed.

The question-answering system categorized questions into six types (usually with typical question elements): (1) **time** questions ("when"), (2) **location** questions ("where"), (3) **who** questions ("who" or "whose"), (4) **what** questions ("what" or "which," used alone or as question determiners), (5) **size** questions ("how" followed by an adjective), and (6) **number** questions ("how many"). Other question types not included above (principally "why" questions or non-questions beginning with verbs "name the ...") were assigned to the **what** category, so that question elements would be present for each question.

Some adjustments to the questions were made. There was a phase of consolidating triples so that contiguous named entities were made into a single triple. Then, it was recognized that questions like "what was the year" or "what was the date" and "what was the number" were not **what** questions, but rather **time** or **number** questions. Questions containing the phrase "who was the author" were converted into "who wrote"; in those with "what is the name of", the triple for "name" was removed so that the words in the "of" phrase would be identified as the principal noun. Other phraseological variations of questions are likely and could be made at this stage.

Once the question type had been determined and the initial set of sentences selected, further processing took place based on the question type. Key elements of the question were determined for each question type, with some specific processing based on the particular question type. In general, we determined the key noun, the key verb, and any adjective modifier of the key noun for each question type. For **who** questions, we looked for a year restriction. For **where** questions, we looked up the key noun in the Macquarie dictionary and identified all proper nouns in all its definitions (hence available for comparison with short answers or other proper nouns in a sentence). For **what** questions, we looked for a year restriction, noted whether the answer could be the object of the key verb, and formed a base set of thesaurus categories for the key noun. For **size** questions, we identified the "size" word (e.g., "far" in "how far"). For **number** questions, we also looked for a year restriction.

3.5.3 Extraction of Short Answers

After the detailed question analysis, processing for each question then examined each selected sentence, attempting to find a viable short answer and giving scores for various characteristics of the sentence. For **time**, **location**, **size**, and **number** questions, it was possible that a given sentence contained no information of the relevant type. In such cases, it was possible that a given sentence could be completely eliminated. In general, however, a data structure for a

BEST COPY AVAILABLE

possible answer was initialized to hold a 50-byte answer and the sentence was assigned an initial score of 1000. An initial adjustment to the score was given for each sentence by comparing the question discourse entities (including subphrases of MWUs) with the sentence discourse entities, giving points for their presence and additional points when the discourse entities stood in the same semantic relation and had the same governing word as in the question.

1. Time Questions - The first criterion applied to a sentence was whether it contained a record that has a TIME semantic relation. The parser labels prepositional phrases of time or other temporal expressions (e.g., "last Thursday"); database records for these expressions were given a TIME semantic relation. We also examined triples containing "in" or "on" as the governing word (looking for phrases like "on the 21st", which may not have been characterized as a TIME phrase) or numbers that could conceivably be years. After screening the database for such records, the discourse entity of such a record was then examined further. If the discourse entity contained an integer or any of its words were marked in the parser's dictionary as representing a time period, measurement time, month, or weekday, the discourse entity was selected as a potential answer.

2. Where Questions - Each sentence was examined for the presence of "in", "at", "on", "of", or "from" as a semantic relation, or the presence of a capitalized word (not present in the question) modifying the key noun. The discourse entity for that record was selected as a potential answer. Discourse entities from "of" triples were slightly disfavored and given a slight decrease in score. If the answer also occurred in a triple as a governing word with a HAS relation, the discourse entity from that triple was inserted into the answer as a genitive determiner of the answer.

3. Who Questions - The first step in examining each sentence looked for the presence of appositives, relative clauses, and parentheticals. If a sentence contained any of these, an array was initialized to record its modificand and span. The short answer was initialized to the key noun. Next, all triples of the sentence were examined. First, the discourse entity (possibly an MWU) was examined to determine the overlap between it and the question discourse entities. The number of hits was then added to all appositives which include the word position of the discourse entity within its span. (A sentence could have nested appositives, so the number of hits can be recorded in multiple appositives.)

The next set steps involved looking for triples whose governing word matched the key verb, particularly the copular "be" and the verb "write". For copular verbs, if the key noun appeared as the subject, the answer was the object, and vice versa. For other verbs, we looked for objects matching the key noun, then taking the subject of the verb as the answer. A test was included here for examining whether the key noun is in the definition, a hypernym, or thesaurus category of the discourse entity, but this was not tested and was removed when the system was frozen.

Another major test of each discourse entity that contained a substring matching the key noun was whether it was modified by an appositive. If this was the case, the appositive was taken as a possible short answer; the discourse entities of the appositive were then concatenated into a short answer. Numerical and time discourse entities were also examined when there was a date restriction specified in the question to ascertain if they could be years, and if so, whether they matched the year restriction. In the absence of a clear sentence year specification, the document date was used.

4. What Questions - The first step in examining the sentences was identical to that of the **who** questions, namely, looking for appositives in the sentence and determining whether a discourse entity had overlaps with question discourse entities. If the key noun was a part of a discourse entity, we would note the presence of the key noun; if this occurrence was in a discourse entity identified as an adjective modifier, the modificand was taken as a short answer and if this short answer was itself a substring of another sentence discourse entity, the fuller phrase was taken as the answer. Similarly, when the key noun was a proper part of a discourse entity and began the phrase (i.e., a noun-noun compound), the remaining part was taken as the short answer.

As with **who** questions, if the key noun was identified as the modificand of an appositive, the appositive was taken as the possible answer. Similarly to **who** questions, we also looked for the copular "be" with the key noun as either the subject or object, taking the other as a possible answer. When the key verb was "have" and the key noun was equal to the object, the subject of "have" was taken as the short answer. In cases like these, we would also insert any adjective modifiers of the noun discourse entities at the beginning of the short answer.

If the key noun was not equal to the discourse entity of the triple being examined, we tested whether the key noun against the DIMAP-enhanced Macquarie dictionary, looking for its presence (1) in the definition of the discourse entity, (2) as a hypernym of the discourse entity, or (3) in the same Macquarie thesaurus category. (For example, in examining "Belgium" in response to the question "what country", where country is not in definition and is not a hypernym, since it is defined as a "kingdom", we would find that "country" and "kingdom" are in the same thesaurus category.) Finally, as with **who** questions, we examined TIME and number discourse entities for the possible satisfaction of year restrictions.

5. Size Questions - For these questions, each triple of a selected sentence was examined for the presence of a NUM semantic relation or a discourse entity containing a digit. If a sentence contained no such triples, it was discarded from further processing. Each numerical discourse entity was taken as a possible short answer in the absence of further information. However, since a bare number was not a valid answer, we looked particularly for the presence of a measurement term associated with the number. This could be either a modificand of the number or part of the discourse entity itself, joined by a hyphen. If the discourse entity was a tightly joined number and measurement word or abbreviation (e.g., "6ft"), the measurement portion was separated out for lookup. The parsing dictionary characterizes measurement words as having a "measures", "unit", "MEASIZE", or "abbr" part of speech, so the modificand of the number was tested against these. If not so present in the parsing dictionary, the Macquarie definition was examined for the presence of the word "unit". When a measurement word was identified, it was concatenated with the number to provide the short answer.

6. Number Questions - The same criterion as used in size questions was applied to a sentence to see whether it contained a record that has a NUM semantic relation. If a selected sentence had no such triples, it was effectively discarded from further analysis. In sentences with NUM triples, the number itself (the discourse entity) was selected as the potential answer. Scores were differentially applied to these sentences so that those triples where the number modified a discourse entity equal to the key noun were given the highest number of points. TIME and NUM triples potentially satisfying year specifications were also examined to see whether a year restriction was met. In the absence of a clear sentence year specification, the document date was used.

3.5.4 Evaluation of Sentence and Short Answer Quality

After all triples of a sentence were examined, the quality of the sentences and short answers was further assessed. In general, for each question type, we assessed the sentence for the presence of the key noun, the key verb, and any adjective qualifiers of the key noun. The scores were increased significantly if these key items were present and decreased significantly if not. In the absence of a clear sentence year specification (for **who**, **what**, and **number** questions containing a year restriction), the document date was used. For certain question types, there were additional checks and possible changes to the short answers.

For **location** questions, where we accumulated a set of proper nouns found in the definition of the key noun, the score for a sentence was incremented for the presence of those words in the sentence. Proper nouns were also favored, and if two answers were found, a proper noun would replace a common noun; proper nouns also present as proper nouns in the Macquarie dictionary were given additional points. Similarly, if a sentence contained several prepositional phrases, answers from "in" phrases replaced those from "of" or "from" phrases. For questions in which the key verb was not "be", we tested the discourse entities of the sentence against the DIMAP-enhanced Macquarie dictionary to see whether they were derived from the key verb (e.g., "assassination" derived from "assassinate").

For **who** and **what** questions, when a sentence contained appositives and in which satisfactory short answers were not constructed, we examined the number of hits for all appositives. In general, we would construct a short answer from the modificand of the appositive with the greatest number of hits. However, if one appositive was nested inside another, and had the same number of hits, we would take the nested appositive. For these questions, we also gave preference to short answers that were capitalized; this distinguished short answers that were mixed in case.

For these two question types, we also performed an anaphora resolution if the short answer was a pronoun. In these cases, we worked backward from the current sentence until we found a possible proper noun referent. As we proceeded backwards, we also worked from the last triple of the each sentence. If we found a plausible referent, we used that discourse entity as the short answer and the sentence in which it occurred as the long answer, giving it the same score as the sentence in which we found the pronoun.

For **size** questions, we deprecated sentences in which we were unable to find a measurement word. We also looked for cases in which the discourse entities in several contiguous triples has not been properly combined (such as number containing commas and fractions), modifying the short answers in such cases.

After scores have been computed for all sentences submitted to this step, the sentences are sorted on decreasing score. Finally, the output is constructed in the desired format (for both 50-byte and 250-byte answers), with the original sentences retrieved from the documents. If a sentence is longer than 250 bytes, the string is reduced based on where the short answer appears in the sentence.

4. TREC-9 Q&A Results

CL Research submitted 4 runs, 2 each for the 50- and 250-byte lengths; the official scores for these runs are shown in Table 1. The score is the mean reciprocal rank of the best answer over all 682 questions that were included in the final judgments. The score of 0.287 for run clr00s1 means that, over all questions, the CL Research system provided a sentence with a correct answer as slightly better than 4th position. This compares to an average score of 0.350 among all submissions for the TREC-9 QA 250-byte answers (i.e., a correct answer slightly better than the 3rd position).

Table 1. CL Research Run Scores

Run	Doc. Num.	Type	Score	TREC Ave.
clr00s1	10	250-byte	0.287	0.350
clr00b1	10	50-byte	0.119	0.218
clr00s2	20	250-byte	0.296	0.350
clr00b2	20	50-byte	0.135	0.218

The CL Research runs differ in the number of documents of the top 50 documents provided by the generic search engine that were processed. As will be discussed below, the number of documents processed reflects a point of diminishing returns in finding answers from the top documents. Table 2 shows the number of questions for which answers were found at any rank for the 682 questions.

Table 2. Answers Found (682)

Run	Doc. Num.	Type	Num	Pct.
clr00s1	10	250-byte	289	0.424
clr00b1	10	50-byte	113	0.166
clr00s2	20	250-byte	296	0.434
clr00b2	20	50-byte	132	0.194

5. Analysis

DIMAP-QA added many components to the system used in TREC-8. The analysis that follows examines the failures of this year's system, along with a description of the incremental steps implemented in dealing with last year's failures. In this way, we hope to capture the characteristics of the question-answering process and the significance of specific components.

As mentioned above, we only processed the top 20 documents provided by NIST. Table 3 clearly indicates that, after the first 10 documents, the amount of incremental improvement from processing more documents is quite small. This table indicates that the CL Research results might better be interpreted in terms of the questions that could possibly have been answered. Table 4 makes these adjustments.

Table 3. Highest ranked top document containing strict answer string

Document Number	Number of Questions
1-10	474
11-20	38
21-30	21
31-40	18
41-50	12
None	130

Table 4. Adjusted scores for documents attempted

Run	Doc. Num.	Type	Score	Adj. Score
clr00s1	10	250-byte	0.287	0.412
clr00b1	10	50-byte	0.119	0.170
clr00s2	20	250-byte	0.296	0.394
clr00b2	20	50-byte	0.135	0.179

The significant difference between the unadjusted and adjusted scores raises an important question: is the question-answering track measuring retrieval

BEST COPY AVAILABLE

performance or question-answering ability? It was noted earlier that the number of semantic relation triples for the questions had declined from 4.5 in TREC-8 to 3.3 in TREC-9. One of these triples contains a question element, so the decline in information content is about one-third. As a result, this year's questions, while being simpler to state, are actually more difficult to answer. This has meant that the likelihood of the retrieval system retrieving a relevant document much less.

While this makes it more difficult for systems relying on the NIST top documents, it also raises the question of what might be an appropriate retrieval strategy. CL Research experimented with the Macquarie dictionary in support of answers to **location** questions (the only "simple" questions in TREC-8, so this strategy was only implemented for that question type in TREC-9). While this strategy may help CL Research performance on other question types, it does not help the retrieval performance shown in Table 3. What it does suggest is that dictionary lookup can usefully be employed in rephrasing a question for retrieving relevant documents. Thus, for example, instead of retrieving birth announcements for "Who is Maria Theresa?", the retrieval engine can search for "archduchess of Austria, queen of Hungary and Bohemia" in addition to "Maria Theresa".

In making improvements to DIMAP-QA for TREC-9, we began by removing many shortcomings noted there (Litkowski, 2000). First, we included documents not processed. Next, we resolved several "bugs", parsing problems affecting both questions and documents and problems in the extraction of semantic relation triples. Dealing with these problems improved the score to 0.550, better than anticipated, but seemingly the best that could be achieved by considering only discourse entities and their relations.

The next stage of development focused on the extraction of short answers. The final result of this process is the set of heuristics described above for the individual question types. We proceeded to this task by categorizing the problems and the likely solutions.

In extending DIMAP-QA to extract 50-byte answers, we found that we could successfully identify appropriate phrases by greater attention to detailed syntactic and semantic structures within the sentence. We looked for opportunities for better characterization of syntactic and semantic roles played by constituents of the sentence; the appositive and genitive determiner constituents led to a significant improvement in performance, particularly for **who** and **what** questions. We were able to exploit this extraction with feedback to the sentence extraction (i.e., when a viable short answer was recognized, its sentence was given a higher score). This extension consisted of question-specific routines for extracting short answers based on the types of semantic relations in which the discourse entities participated. This improved scores to 0.740 for sentences and 0.493 for short answers.

At this point in development, it became clear that the model we were implementing could be characterized as just-in-time: improvements could be attained by implementing slight refinements taken from techniques like named-entity extraction and query expansion. It was only at this point in development that the lexical resources were integrated. Although these resources could have been used directly to answer questions, the just-in-time model used them instead for substantiation. For example, in "where" questions, definitions provided a background set of discourse entities used in evaluating document sentences. For "what" questions (e.g., "what country"), dictionary definitions were examined to determine whether a document discourse entity was defined as or had the hypernym "country". If no match, the thesaurus was examined to determine if the hypernym for a document discourse entity was in the same thesaurus category (e.g., as "country" where "Belgium" is defined as a "kingdom").

The final set of improvements to DIMAP-QA came from a more detailed evaluation of the short answers. These changes can be characterized as reflecting a more global view of the questions, identifying their critical components and implementing procedures for decreasing the scores of sentences that were given inappropriately high scores.

Incorporation of the lexical resources and the further evaluation of the short and sentence answers in light of the key words in the questions improved the TREC-8 scores to 0.803 for sentences and 0.597 for short answers. It was at this point that the system was frozen for the participation in TREC-9.

In examining the TREC-9 results, we have taken a similar approach to categorizing the failures. In general, we have found that there is nothing qualitatively different from our performance with the TREC-8 questions. We have, for the most part, extracted appropriate sentences for detailed analysis

(96.5%). Availability of the appropriate document is the most prevalent problem (34% of the failures). About 12% of the failures can be attributed to the need to degrade the scores of too highly scored sentences. Another 10% require improved characterization and extraction of constituents from the parse output. About 10% of the questions can be answered by improved routines for interacting with the lexical resources. About 6% can be characterized as difficult problems. The remaining problems seem to require better examination of the question components or modification of the algorithms for the individual questions. The routines that were implemented for the specific question types need to be evaluated for how well they work together (i.e., as some routines were implemented, they may have degraded other routines).

As mentioned earlier, we experienced significant problems with processing *Associated Press*, *Wall Street Journal*, and *San Jose Mercury News* documents. We reran the entire 10- and 20-document sets after our formal submission and estimate that these problems reduced our overall performance by about 0.028.

In Table 4, the adjusted score for the 20-document run was 0.394, compared to 0.412 for the 10-document run. This indicates that we actually experienced a degradation in performance in going from 10 to 20 documents. Overall, in examining the official scores, looking for cases where we performed better on the 10 document set than the 20 document set, we found that this amounted to 0.042 loss of points.

6. Anticipated Improvements

The immediate possibilities for improvements are many and the possibilities for exploration are quite diverse. In addition, there are opportunities to be explored for integrating DIMAP-QA within more generalized search engines.

The clearest avenue of improvement is indicated by the question variations in questions 701 to 893. For 16 variants we were unable to answer in the base 500 questions because the appropriate documents were not in the top 10; the problem persisted for 8 questions. Of the remainder, we were able to obtain an answer under 2 variations. Of the other 38 variation sets, we did not obtain answers for 18 of the base questions, but were able to find answers in one or more of the variations for 11 sets. This suggests that improvements may be obtained by finding the "best" canonical form for a question. (For most of the variants, the reformulated questions gave rise to quite different document positions of appropriate documents, underscoring again the significance of the retrieval problem.)

The use of the dictionary and thesaurus in this year's system was quite rudimentary. Analyzing the questions, we found that 35% were either definitional, answerable by dictionary lookup, or supportable by the dictionary. Implementing procedures similar to those used in answering **where** questions will lead to substantial improvements.

7. Summary

The CL Research system was reasonably successful in answering questions by selecting sentences from the documents in which the answers occur. The system generally indicates the viability of using relational triples (i.e., structural information in a sentence, consisting of discourse entities, semantic relations, and the governing words to which the entities are bound in the sentence) for question-answering. Post-hoc analysis of the results suggests several further improvements and the potential for investigating other avenues that make use of semantic networks and computational lexicology.

Acknowledgements

We want to thank Richard Tardif of Macquarie for making Big Mac and the Macquarie thesaurus available, Steve Green for his perl scripts for converting the Macquarie thesaurus into WordNet format, and Thomas Pötter for directing us to MetaKit and making us think hard about our approach.

References

Fellbaum, C. (1998). *WordNet: An electronic lexical database.* Cambridge, Massachusetts: MIT Press.

Green, S. J. (Stephen.Green@East.Sun.COM). (2000, 27 January). (Macquarie thesaurus).

Litkowski, K. C. (2000). Question-Answering Using Semantic Relation Triples. In: Voorhees, E. M. & Harman, D. K. (eds) *The Eighth Text Retrieval Conference (TREC-8)*, NIST Special Publication 500-246, 349-356.

Richardson, S. D. (1997). Determining similarity and inferring relations in a lexical knowledge base [Diss], New York, NY: The City University of New York.

ACSys/CSIRO TREC-9 Experiments

David Hawking[*]
CSIRO Mathematics and Information Sciences,
Canberra, Australia
David.Hawking@cmis.csiro.au

February 7, 2001

Unfortunately, ACSys was able to commit very few resources to TREC experiments this year and participated only in the Web track. I took the retrieval component (PADRE99) of our standard metadata/content Intranet search engine, modified it to handle TREC-formatted web pages and ran the unexpanded topic titles as queries. I also generated a set of manual (non-interactive) queries from the topic statements.

It transpired that the query "angioplast7" generated by the automatic title-only process retrieved 0 documents and these runs were rejected by the NIST submission process. Therefore, only the manual run (`acsys9mw0`) was officially assessed. It achieved an average precision of 0.2486.

As in TRECs 6, 7 and 8 [Hawking et al. 1997], the basic relevance scoring method used in official ACSys adhoc runs the Okapi BM25 weighting function [Robertson et al. 1994]

$$w_t = q_t \times tf_d \times \frac{\log(\frac{N-n+0.5}{n+0.5})}{2 \times (0.25 + 0.75 \times \frac{dl}{avdl}) + tf_d} \tag{1}$$

where w_t is the relevance weight assigned to a document due to query term t, q_t is the weight attached to the term by the query, tf_d is the number of times t occurs in the document, N is the total number of documents, n is the number of documents containing at least one occurrence of t, dl is the length of the document and $avdl$ is the average document length (measured either in bytes or in indexable words). A slight modification has been applied to ensure that weights are never negative.

As in TREC-8 runs using PADRE99, the index included stopwords, numbers and strings comprised of letters and digits. Words in the index were not stemmed but query-time stemming and truncation operators were available. In the official (manual) run, no automatic expansion or relevance feedback was employed.

The queries listed on the next page were processed as follows. In general, each query element (word, stem, phrase or dysjunctive group) was treated as a constraint. Regardless of BM25 score, documents matching more constraints are presented ahead of documenents matching fewer constraints. This results in a tiered ranking. Within each tier, documents are ranked by BM25 score.

---

[*] The author wishes to acknowledge that this work was carried out within the Cooperative Research Centre for Advanced Computational Systems established under the Australian Government's Cooperative Research Centres Program. Despite this, the work was actually carried out in Greece, while attending a Summer School and a Conference.

Example	Feature	Notes
"babe ruth"	Phrase	
theater#"	Stem operator	
[pine pinus]	Dysjunctive group	Constraint satisfied by either word.
~grow	Non-constraint operator	Word contributes to score but is not a constraint.
recurr*	Truncation operator	Matches any word starting with recurr

```
451 Bengal [cat cats]
452 [beaver beavers] [salt saline fresh saltwater freshwater sea seawater estuary]
453 [hunger famine malnutrition starvation starving] [relief aid feed*]
454 ["parkinson's disease" parkinsonism]
455 "jackie robinson" ["first game" "first appearance" "debut"] [year season]
456 ["world end" "end of the world" apocalypse] 2000 [group sect cult religion followers disciples]
457 chevrolet [truck trucks]
458 [fast fasting] [religion islam muslim jewish judaism catholic*]
459 [foreclose foreclosed foreclosure]
460 Moses ~[israel* prophet "red sea" "burning bush" commandments egypt pharoah]
461 "lava lamp#"
462 ["real estate" realty realtor "estate agent#"]["new jersey" NJ] ~[house houses appartment# residen*]
463 [tartan tartans plaid plaids] ~[scottish clan clans]
464 "nativity scene*" [ban bans banning banned]
465 [disease diseases syndrome] [deer venison]  ~[lyme tick ticks]
466 "peer gynt" ~[compose grieg composer suite folk folklore]
467 [dachshund dachshunds "wiener dog"] ~[breed breeder pedigree]
468 incandescent [light bulb# globe# filament tungsten] [carbon edison swann invent*]
469 steinbach nutcracker*
470 mistletoe [beneficial benefit health medical medicine drug nutrition feed food sustain cure]
471 "mexican food" [australia* europe* asia asian afric* spain philipines]
472 antique appliance restor* ~[dealer* collect* museum*]
473 "toronto film" [award won winner winning prize]
474 ["e mail" email e-commerce] [profit# turnover sales money] ~[internet online dotcom]
475 zirconium [melting boiling hardness valency *valent react reaction ductil* refractory malleable "atomic number"]
476 "jennifer aniston" [movie movies cinema tv television]
477 [rccl "royal carribean"] [ship ships liner liners vessel vessels]
478 baltimore mayor*
479 "kappa alpha psi" [fraternity sorority college# university#]
480 [traffic roads cars interstate vehicles] [washington DC maryland virginia arlington bethesda beltway]
481 "babe ruth" 1920
482 "growth rate" [pine pinus pinetree pines] ~[grow growing]
483 [rosebowl "rose bowl"] parade
484 [auto car cars vehicle#] skoda
485 [gps "global positioning"] clock accuracy
486 [eldorado "el dorado"] reno ~[nevada casino street road]
487 angioplasty [repeat follow-up recurr*]
488 "newport beach" [california ca] [entertain* movies arcade* sport# game games theater#]
489 calcium [diet dietary supplement# health osteoporosis bone]
490 [motorcycle motorbike] helmet# [law legal violation compulsory]
491 ["japanese wave" tsunami] ~[damage killed died property destroyed "swept away" relief aid]
492 "savings bond#" ~[us federal interest maturity rate* denominat*]
493 [retirement communit*]
494 nirvana ~[guitar* drum* lyrics penned songs "rock group" "rock band"]
495 twenties
496 [tmj "temporal mandible"] [syndrome condition suffer*] ~[treatment therapy symptom* cause* "due to"]
497 [orchids "orchid growing"]
498 "hair transplant#" ~[follicle* bald* scalp]
499 "pool cue"
500 ["dna test*" "dna analysis"] ~[forensic police fbi investigat* dna]
```

BEST COPY AVAILABLE

Bibliography

HAWKING, D., THISTLEWAITE, P., AND CRASWELL, N. 1997. ANU/ACSys TREC-6 experiments. In E. M. VOORHEES AND D. K. HARMAN Eds., *Proceedings of TREC-6* (Gaithersburg MD, November 1997), pp. 275–290. NIST special publication 500-240, `http://trec.nist.gov`.

ROBERTSON, S. E., WALKER, S., HANCOCK-BEAULIEU, M., AND GATFORD, M. 1994. Okapi at TREC-3. In D. K. HARMAN Ed., *Proceedings of TREC-3* (Gaithersburg MD, November 1994). NIST special publication 500-225.

Melbourne TREC-9 Experiments

Daryl D'Souza Michael Fuller
James Thom Phil Vines Justin Zobel
Department of Computer Science, RMIT University
GPO Box 2476V, Melbourne Victoria Australia 3001
{djds, msf, jat, phil, jz}@cs.rmit.edu.au

Owen de Kretser
Department of Computer Science and Software Engineering
The University of Melbourne, Victoria 3010, Australia
oldk@cs.mu.oz.au

Ross Wilkinson Mingfang Wu
CSIRO, Division of Mathematics and Information Science
723 Swanston St., Carlton VIC 3053
{Ross.Wilkinson, MingFang.Wu}@cs.mu.oz.au

February 1, 2001

1 Introduction

We report results for experiments conducted in Melbourne—at CSIRO, RMIT, and The University of Melbourne—for TREC-9. We present results for the interactive track, cross-lingual track, main web track, and the query track.

2 Interactive Track

2.1 Introduction

We have been continuously investigating technologies for delivering retrieved documents to support interactive question answering. In this year's interactive track, we focused on the role of a document surrogate in the interactive fact finding task. In this experiment, we compared two types of document surrogates

Submitted to the 9th Text Retrieval Conference (TREC-9), Gaithersburg, MD, USA, November 13–16, 2000.

in the two experimental systems. One system uses the document title and the first 20 words of a document as the document's surrogate, while the other system uses the document title and the best three Answer Indicative Sentences extracted from the document as the document's surrogate. The results show that subjects can find significantly more facts from the system using 3 sentences than from the other system.

2.2 Hypothesis

This year's track reflects two of the major characteristics of interactive information searching: the questions are concentrated on the fact finding, and the time for answering each question is very short (5 minutes). As an average reader can only scan a limited number of words within 5 minutes, the challenge is how to help the user to locate the facts or find the documents that may contain the facts while reading the limited number of words.

For most web search engine (e.g. Altavista, Excite), this has been achieved by displaying the surrogate of a document, which mainly includes the title and the first *N* words from the document. The purpose of the surrogate is to indicate the main theme of the document. This kind of surrogate may be more suitable for the learning and exploration types of information needs, but less suitable for fact finding type of information need. Based on our pilot investigation into the interactive fact finding task, we observed that:

- The relevant facts may exist within a small chunk of documents, and this small chunk may be not necessarily related to the main theme of the document.
- This small chunk usually contains the keywords, and is in the form of a complete sentence. We call this sentence the answer indicative sentence (AIS).
- When a user is scanning through a document to search for facts, s/he usually tries to locate an answer indicative sentence by looking around the query keywords, and therefore either find the facts, or decide whether to read the document further or discard it.

Our hypothesis is that the above-mentioned answer indicative sentences should provide a better surrogate of the document than the first *N* words, for the purpose of interactive fact finding. Therefore our experiment focused on the comparison and evaluation of two systems using different surrogates. The control system *First20* uses the title and the first twenty words as the surrogate of a document, and the test system *AIS3* uses the title and best three answer indicative sentences as the surrogate of a document. The performance was measured by the effectiveness of each system in helping to locate answer facts, users' subjective perception of the systems, and the effort required by users to locate answers.

2.3 System Description

Both systems in this experiment use the `mg` [4] search engine for indexing and retrieval. The two systems provide natural language querying [2] only. For each query, both systems present a user with the surrogates of the top 100 retrieved documents in 5 consecutive pages, with each page containing 20 document surrogates. Each system has a main window for showing these surrogate pages. A document reading window is popped up when a document surrogate is clicked. If a user finds a fact from the document reading window, s/he can click the "Save Answer" button in this window and a save window will be popped up for the user to input the newly found fact or modify previously saved facts.

The difference between two systems is what is presented on their main windows. The main window of the control system (*First20*) is shown in Figure 1. This kind of presentation is quite similar to those web search engines such as Altavista and Excite. The main window of the test system (*AIS3*) is shown in Figure 2. Roughly, the number of words on each page of an *AIS3* window is three times that of the *First20* window. Also, there is a *save* icon next to each answer indicative sentence, with the same function as the "Save Answer" button in the document reading window. If a user finds a fact from the sentence, s/he can save the fact directly by clicking this icon.

The three best AIS are dynamically generated after each query search according to the following procedure:

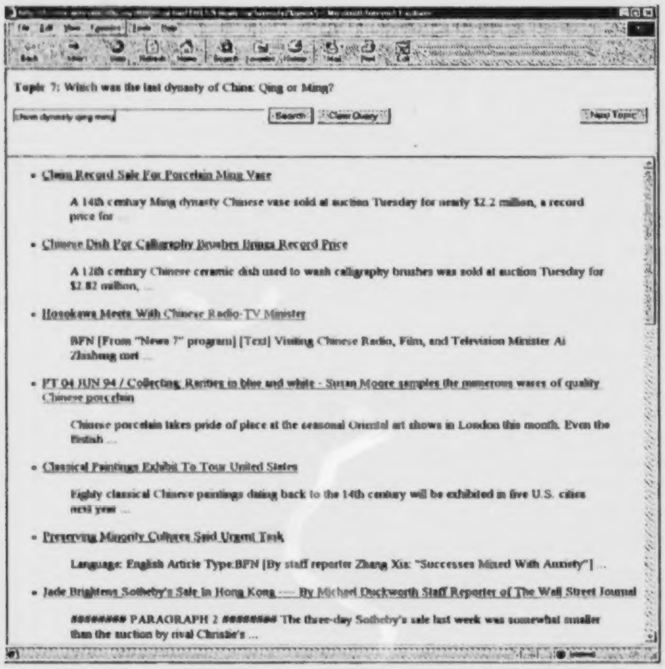
- An AIS should contain at least one query word and be at least ten words long.
- The AISs are first ranked according to the number of unique query words contained in each AIS. If two AISs have the same number of unique query words, they will be ranked according to order in which they occur within the document.
- The top three AIS are then selected.

2.4 Experiment

2.4.1 Procedure

Each subject searched eight topics according to the TREC-9 Interactive Track experimental guidelines, with four topics on each system. During the experiment, the subject followed the following steps:

- Reading the introduction to the experiment.
- Filling in the Pre-Search Questionnaire.
- Demonstration of main functions of each system.
- Hands on practice with both systems.



Topic 7: Which was the last dynasty of China: Qing or Ming?

china dynasty qing ming

- Claim Record Sale For Porcelain Ming Vase

  A 14th century Ming dynasty Chinese vase sold at auction Tuesday for nearly $2.2 million, a record price for ...

- Chinese Dish For Calligraphy Brushes Brings Record Price

  A 12th century Chinese ceramic dish used to wash calligraphy brushes was sold at auction Tuesday for $2.82 million, ...

- Hosokawa Meets With Chinese Radio-TV Minister

  BFN [From "News 7" program] [Text] Visiting Chinese Radio, Film, and Television Minister Ai Zhisheng met ...

- FT 04 JUN 94 / Collecting: Rarities in blue and white - Susan Moore samples the numerous wares of quality Chinese porcelain

  Chinese porcelain takes pride of place at the seasonal Oriental art shows in London this month. Even the British ...

- Classical Paintings Exhibit To Tour United States

  Eighty classical Chinese paintings dating back to the 14th century will be exhibited in five U.S. cities next year ...

- Preserving Minority Cultures Said Urgent Task

  Language: English Article Type:BFN [By staff reporter Zhang Xia: "Successes Mixed With Anxiety"] ...

- Jade Brightens Sotheby's Sale In Hong Kong ---- By Michael Duckworth Staff Reporter of The Wall Street Journal

  ######## PARAGRAPH 2 ######## The three-day Sotheby's sale last week was somewhat smaller than the auction by rival Christie's ...

Figure 1: The main window of the *First20* system.

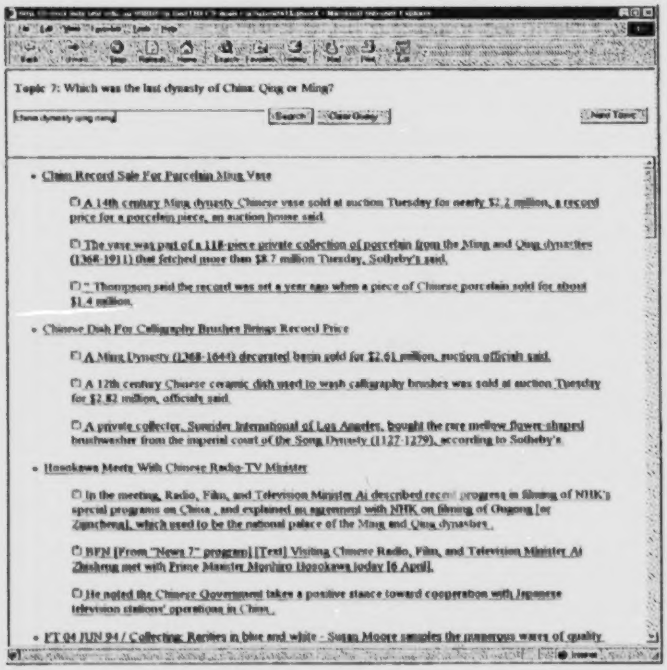
- Search four topics on each system with Pre-Search questionnaire and Post-search questionnaire per topic, and a Post-System questionnaire per system.
- Filling in exit questionnaire.

It takes about 1.5 hours for a subject to finish the whole procedure.

2.4.2 Subjects

Sixteen paid subjects were recruited via an RMIT internal university newsgroup. There are five females and eleven males. The average age of sixteen subjects is 23, with the youngest 19 and oldest 39. They have 5.1 years online search experience on the average. All subjects are the students from the Department of Computer Science, nine of them are undergraduate students, the other seven subjects already had a Bachelor degree and are studying for a higher degree (3 on graduate diploma and 4 on master degree).

BEST COPY AVAILABLE



Topic 7: Which was the last dynasty of China: Qing or Ming?

china dynasty qing ming

- China Record Sale For Porcelain Ming Vase
  - ☐ A 14th century Ming dynasty Chinese vase sold at auction Tuesday for nearly $2.2 million, a record price for a porcelain piece, an auction house said.
  - ☐ The vase was part of a 118-piece private collection of porcelain from the Ming and Qing dynasties (1368-1911) that fetched more than $8.7 million Tuesday, Sotheby's said.
  - ☐ " Thompson said the record was set a year ago when a piece of Chinese porcelain sold for about $1.4 million.
- Chinese Dish For Calligraphy Brushes Brings Record Price
  - ☐ A Ming Dynasty (1368-1644) decorated basin sold for $2.61 million, auction officials said.
  - ☐ A 12th century Chinese ceramic dish used to wash calligraphy brushes was sold at auction Tuesday for $2.82 million, officials said.
  - ☐ A private collector, Sunrider International of Los Angeles, bought the rare mellow flower-shaped brushwasher from the imperial court of the Song Dynasty (1127-1279), according to Sotheby's.
- Hosokawa Meets With Chinese Radio-TV Minister
  - ☐ In the meeting, Radio, Film, and Television Minister Ai described recent progress in filming of NHK's special programs on China , and explained an agreement with NHK on filming of Gugong [or Zijincheng], which used to be the national palace of the Ming and Qing dynasties .
  - ☐ BFN [From "News 7" program] [Text] Visiting Chinese Radio, Film, and Television Minister Ai Zhisheng met with Prime Minister Morihiro Hosokawa today [6 April].
  - ☐ He noted the Chinese Government takes a positive stance toward cooperation with Japanese television stations' operations in China .
- FT 04 JUN 94 / Collecting: Rarities in blue and white - Susan Moore samples the numerous wares of quality

Figure 2: The main window of the *AIS3* system.

2.4.3 Data Collection Methods

Transaction logging and questionnaires were used to collect data. During the experiment, every significant event—such as documents read, facts saved and their supporting documents, and queries sent—were logged and time-stamped automatically. The questionnaires used were the standard questionnaires used by participants in the Interactive Track.

2.5 Evaluation

2.5.1 System

Effectiveness

The effectiveness of the each system is evaluated by the number and the quality of the saved answers. There are two types of topics in this year's interactive track. Type 1 topics are of the form "find any *n* Xs". Type 2 topics are of the form "compare two specific Xs". For the Type 1 topics (topic 1-4), a complete

BEST COPY AVAILABLE

answer consists of n facts. For the Type 2 topics (topic 5-8), two facts are usually needed to make the comparison. We observed that only for topic 7 (Type 2), the answer may sometimes be supported by only one fact.

The saved facts and the saved documents were sent to the NIST for judgement. For each search session, two judgements were made: whether the subject found the required number of facts (for topics of Type 1) or whether the subject answered the question correctly (for topics of Type 2); and whether the saved facts (or answers) are supported by the saved documents. Both judgements have three scores: all, some, or none.

A fully successful session is defined as whether the question is fully answered and whether the answer is fully supported (i.e. the both judgments are "all"). If we give a score of '1' to such a successful session and a score of '0' to any other sessions, then there are 14 successful sessions in total for users of *First20* and 27 successful sessions in total for users of *AIS3*. The difference between the two systems is significant at level 0.01 (two tailed t-test).

Table 1 shows the successful sessions topic by topic. We can see that users of *AIS3* has more successful sessions for all topics except the Topic 3.

Topic	1	2	3	4	5	6	7	8	Total
First20	0	0	0	2	3	3	5	1	14
AIS3	1	2	0	3	4	7	8	2	27

Table 1: The number of the fully successful sessions per topic.

Table 2 shows the number of the fully successful sessions subject by subject. We see that of the sixteen subjects, ten subjects had more successful sessions when using *AIS3* than when using *First20*; only two subjects had more successful sessions when using *First20* than when using *AIS3*.

Subject	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
First20	0	0	1	1	1	2	1	3	2	0	0	1	1	0	0	1
AIS3	2	2	1	1	3	2	2	1	1	1	1	1	2	3	2	2

Table 2: The number of the fully successful sessions per subject.

Although the subjects may not get the full answer in some sessions, the subjects sometimes demonstrated the ability to find a partial answer. We need not simply classify these sessions as failure, but instead may consider them as partially successful sessions. Therefore, we can award an adjusted score in the range $[0, 1]$. For each topic, each fact that is correctly identified and supported by a document contributes $1/n$ toward the score, where n is the number of required facts for the topic. Overall, *AIS3* gets score 0.65 and *First20* 0.47; the difference between the two systems based on the adjusted score is also statistically significant, at level 0.03 (two tailed t-test).

Table 3 shows the average score across subjects per topic for each system. *AIS3* has the higher score than *First20* for all topics except for the topic 5.

Topic	1	2	3	4	5	6	7	8	Mean
First20	0.08	0.00	0.38	0.71	0.81	0.75	0.75	0.31	0.47
AIS3	0.46	0.25	0.47	0.79	0.75	0.94	1.0	0.56	0.65

Table 3: The comparison of two systems topic by topic based on adjusted scores.

Based on the above results, the hypothesis that the AIS is better document surrogate than the *First20* for fact finding task is supported.

Perception of the system

The Subjects' perception of the systems is captured from three questions in exit questionnaire. The three questions are:

- Question 1: Which of the two systems did you find easier to learn to use?
- Question 2: Which of the two systems did you find easier to use?
- Question 3: Which of the two systems did you like the best overall?

The distribution of subjects' choice is shown in Table 4. We can see that: for question 1, 17% subjects selected *First20* while 50% subjects selected *AIS3*. For question 2, 25% subjects selected *First20* while 69% subjects selected *AIS3*. For question 3, 31% subjects selected *First20* while the other 69% selected *AIS3*. This suggests that the subjects preferred the *AIS3* system.

Questions 1 and 2 were also asked in the post system questionnaire. However instead of asking subjects to compare the two systems, the subjects were asked to judge the systems independently on a 5-point Likert scale. There is not much difference between the two systems on learning effort (*First20*: Mean = 4.0, *AIS3*: Mean = 4.1). The difference between the two systems on user perception of usefulness is statistically significant (*First20*: Mean = 3.5, *AIS3*: Mean = 4.1. two tailed, paired t-test $p < 0.03$). The result for "easy" was unexpected: we thought that the main window of *First20* was simpler than that of *AIS3*, thus would be easier to use than *AIS3*. The subject's selection and judgement may have been influenced by how well they felt they had completed their tasks.

	Easier to learn	Easier to use	Liked the best overall
First20	3	4	5
AIS3	8	11	11
No difference	5	1	

Table 4: Subjective comparison of two systems.

2.5.2 User Effort

The effort required of subjects to determine answers for each topic can be measured in terms of the number of documents they read, the number of title pages viewed, and the number of queries sent for each topic. On the average, the subjects read fewer documents and fewer title pages, and sent fewer queries from *AIS3* than from *First20*, as shown in Table 5. The difference is statistically significant at level 0.01, 0.001, and 0.02 respectively (two tailed t-test). This may not necessary mean that the users of *AIS3* took less effort than the users of *First20*, as the main page of *AIS3* displayed more text than that in *First20*. However, this may indicate that the extracted answer indicative sentences of *AIS3* may have helped subjects to find the answer or find the documents where the answer may be found.

	First20	*AIS3*
	Mean(SD)	Mean(SD)
Number of documents read	3.42(1.22)	2.66(0.77)
Number of pages viewed	2.80(1.64)	1.98(0.97)
Number of unique queries sent	2.14(0.56)	1.73(0.57)
Number of terms per query	3.25	3.26

Table 5: Subject's interaction with the systems.

2.5.3 Perception of the Topics

Before each search, subjects were asked about their familiarity about the topic. As show in the Table 6, overall, subjects have low familiarity with all topics (all under 3 on a 5-point Likert scale). Of eight topics, Topic 7 had the highest familiarity. Nine subjects claimed that they knew the answer before the search, but four of these subjects were wrong. After the search, three of these four subjects got the right answer.

Topic	1	2	3	4	5	6	7	8
First20	1.75	1.25	1.63	1.38	1.38	1.25	2.63	1.38
AIS3	1.25	1.75	1.50	1.25	1.75	1.00	2.00	1.50
All	1.50	1.50	1.56	1.31	1.56	1.13	2.31	1.44

Table 6: Average score of subject's familiarity with each topic.

After each topic, subjects were asked about their satisfaction with the search results and certainty about the answer. Generally, the users of *AIS3* had higher satisfaction and certainty (satisfaction: *First20* Mean = 3.16, *AIS3* = 3.56; certainty: ¡S-Del¿*First30* Mean = 3.50, *AIS3* = 3.89), but these differences are not significant.

There is no significant correlation found between the familiarity and the number of successful sessions, the satisfaction, or the certainty.

Most tested topics were very clear to the subjects. There are two topics which had different interpretations among subjects. One is the Topic 1: "What are the names of three US national parks where one can find redwoods?" — many subjects saved state parks. Another is Topic 2: "Identify a site with Roman ruins in present day France?" — some subjects were not certain about the area scope of the site. Some subjects said they did find Southern France, but they did not think that could be counted as an answer, instead trying to find the specific name of the site.

2.5.4 Subjects Difference

It is interesting that our subjects fall into two groups: one whose first language is English, and another whose second language is English and their first language varies. The subjects of the latter type are all international students from Asia. The native language group has 7 subjects while the foreign language group 9 subjects.

We break the subjects into two groups and summarise the data accordingly for each system based on the adjusted scores, the result is as shown in Table 7. No difference is detected between two groups of each system. This may indicate that the language and culture background are unlikely to have influenced subjects' performance for the tested topics.

	Native	Foreign
First20	0.46	0.49
AIS3	0.66	0.64

Table 7: The comparison of two groups (native language and foreign language) based on the adjusted score.

2.6 Discussion

The experiment investigated the role of document surrogate in the interactive fact finding task. The experiment results show that using an AIS3 as a document surrogate is significantly better than the First20 in helping users locate relevant documents and thus find more relevant facts. Subjects also more preferred the *AIS3* system than the *First20* system.

Although our hypothesis has been supported in the experiment, we understand that more topics and wider variety of document collections will need to be tested to further validate the hypothesis.

3 Cross-Lingual Track

This year we participated in the Chinese-English cross-lingual track, drawing on our experience from our involvement in the Chinese track several years ago. We used two approaches to convert the problem to one of monolingual retrieval. First, we tested converting the English language queries to Chinese (run *rmitcl002*), and second, we tested converting the Chinese document collection to English (run *rmitcl003*). In both approaches we made use of the online dictionaries that were made available.

The translations were on a word by word basis. For the English-to-Chinese translation, if a word that contained uppercase letters was not in the dictionary, we converted it to lower case and tried again. The reason for this is that some proper nouns appear in the dictionary with a capitalised first letter, however for words at the start of a sentence it is more appropriate to convert to lower case.

We also tested combination of evidence (*rmitcl001*), combining the results of the two previous runs based on normalised similarity values, that is, $sim_{new} = 0.5 \times sim'_1 + 0.5 \times sim'_2$, where sim'_1, and sim'_2 are the normalised similarity measures from two runs above. We also included a monolingual run as a baseline.

Our monolingual run results were somewhat lower than the median. We are not completely sure why this is the case but suspect it was partly because the run was a straight processing of the data with no special treatment, and because character indexing rather than word indexing was used. Unfortunately the cross language runs produced random results; there is obviously a problem with the software which we are working to resolve.

4 Main Web Track

Four runs were submitted, labelled `rmitWFGweb`, `rmitWFLweb`, `rmitNFGweb` and `rmitNFLweb`. These correspond to two categories of indexes and, in each case, to two filtering protocols. The index categories were *global* (G) and *local* (L), both based on the wt10g corpus. The global index centrally-indexed all documents; the local indexes were based on five, separate subsets of the data source, as per distribution across 5 wt10g CDs. Each of the two index cases were further classified according to the filtering protocols, *no filter* (NF) and *with filter* (WF). Thus, `rmitWFGweb` refers to the filter-based, global index run.

This section is structured as follows. Section 4.1 presents the similarity ranking formulation to score and subsequently rank the documents. Details of the two filtering protocols are presented in Section 4.2. After indexing Title-only fields of TREC topics 451 to 500 were used in the querying process. Manual queries were used, as discussed in Section 4.3. We used tools from the `mg` system [4] to construct and query indexes. Document sources were stopped and stemmed during the indexing process, and so too were queries, prior to submitting them for ranking of documents. Retrieval effectiveness results for various runs are presented in Section 4.4.

4.1 Relevance scoring method

The combining function used to establish similarity of documents and queries was the standard Cosine measure [5], where similarity, $S_{q,d}$ between document d and query q is given by:

$$S_{q,d} = \frac{\sum_{t \in q \cap d}(w_{q,t} \cdot w_{d,t})}{W_q \cdot W_d}$$

where the document-term and query-term weights are computed, respectively, as:

$$w_{d,t} = log_2(f_{d,t} + 1)$$

$$w_{q,t} = log_2(f_{q,t} + 1) \cdot log_2\left(\frac{N}{f_t} + 1\right)$$

The terms $f_{x,t}$ $(x = q|d)$, f_t and N are, respectively, frequency of term t in x, number of documents containing t, and the total number of documents. Finally, W_x is given by

$$W_x = \sqrt{\sum_{t=1}^{n} w_{x,t}^2}$$

for the n terms in the vocabulary.

4.2 Filtering versus non-filtering

We used two term extraction protocols in the indexing process. For the NF cases the default term extraction policy used by the indexing tool was used. Words are extracted as follows:

- A word is a string of alphanumeric characters delimited by non alphanumeric or space symbols.
- Long digit strings are truncated at every fourth byte, until a non-digit is encountered. Each truncated portion constitutes a word, including the residue, if any.
- Words in tags are ignored.

In the WF cases, data sources were subjected to a filtering process prior to indexing:

- A word is a string of alphanumeric characters delimited by non alphanumeric or space symbols; a word must begin with a letter and may not have more than two digits.
- Words from a long, non-space-delimited string are not extracted beyond the tenth character in the original string.
- Words in tags are ignored, except words inside HTML comments.

Topic-id	Word(s) before change	Word(s) after change
455	`whan`	`when`
463	`tartin`	`tartan`
464	`nativityscenes`	`nativity scenes`
474	`bennefits`	`benefits`
475	`compostion`	`composition`
477	`Carribean`	`Caribbean`
483	`rosebowl`	`rose bowl`
487	`angioplast7`	`angioplasty`

Table 8: *Summary of amendments TREC Topics 451-500.*

4.3 Queries

Title-only fields of TREC Topics 451 to 500 were used. These were manually amended to ensure that at least one document was ranked for each run and query, and to correct spelling inconsistencies between query terms in Title and other fields. Table 8 summarises these changes; it presents the part of the query (prior to stopping and stemming) that was modified.

Note that amendments in the first and last table entries are inconsequential. Prior to indexing `when` is removed because it is a stop word and `whan` is unlikely to appear in the document text. Similarly, `angioplasty` is stemmed to `angioplast`; leaving `angioplast7` as is would cause both indexing protocols to index the term `angioplast`.

4.4 Results

Unfortunately, after submission, the WF result runs were identified as being flawed, due to an erroneous word filter. Nevertheless, corrected runs, while depicting improved performances, revealed that word filtering did not improve performance of the WF cases over the NF cases. The filtering process was motivated by the rationale that removal of URL references and inclusion of words in comments would improve overall performance; this was not the case.

Figures 3 and 4 present the Recall-Precision performances of NF versus WF for the global and local scenarios, respectively. The three-way relevance judgements (not relevant, relevant, highly relevant) were altered to reflect a binary relevance (relevant, not relevant) by re-codifying *highly relevant* documents as *relevant*.

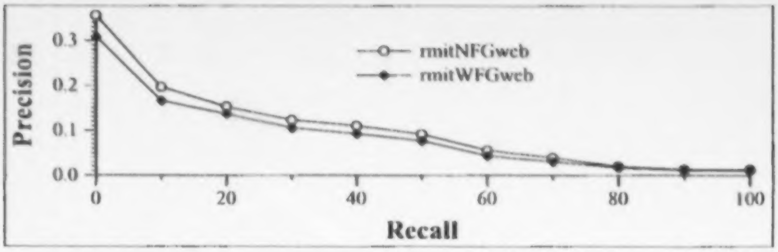


Figure 3: *Main web track: Retrieval P-R performances of global indexes.*

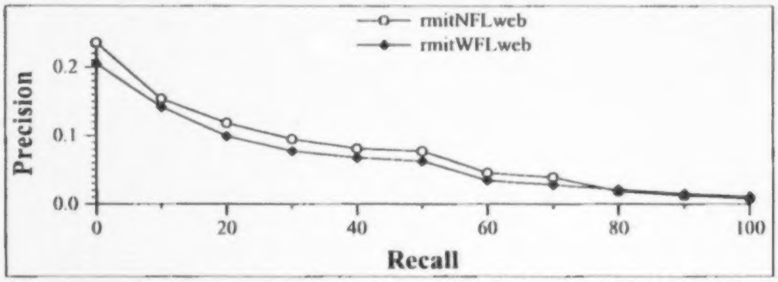


Figure 4: *Main web track: Retrieval P-R performances of local indexes.*

5 Query Track

5.1 Stage 1: Query Variations

Three variations of the TREC Topics 51–100 were manually created:

- `UoM1a`: A 2–3 word query based on the topic statement.
- `UoM1b`: Another 2–3 word variation based on the topic statement.
- `UoM2`: A sentence based on the topic and relevance judgements.

All query variations were created by the same person and roughly 2–3 minutes were spent on each topic for each variation.

5.2 Stage 2: Retrieval Variations

There were 43 different query sets made available by the participants. Prior to retrieval runs, each topic of each query variation had stopwords removed. Two different retrieval systems were used, each based on the full-text retrieval system mg [4].

The first system used was the standard document-based version of `mg` using the following vector-space similarity measure with a normalised-by-maximum-frequency variant of the query-term weights:

$$S_{q,d} = \frac{\sum_{t \in T_{q,d}} (w_{q,t} \cdot w_{d,t})}{W'_d} \tag{1}$$

where

$$\begin{aligned} w_{d,t} &= \log_e(f_{d,t}) + 1 \ , && (2) \\ w_{q,t} &= \log_e(\frac{f^m}{f_t} + 1) \cdot (\log_e(f_{q,t}) + 1) \ , \text{ and} && (3) \\ W'_d &= (1 - s) + s \cdot \frac{W_d}{W_{av}} \ . && (4) \end{aligned}$$

Document length normalisation is by *pivoted cosine normalisation* [3] where W_{av} is the average W_d over all d, and s, the slope of the pivoted cosine normalisation function, is taken to be 0.7. Using the Q-expression notation developed by Zobel and Moffat, this formulation is expressed as `BD-ACI-BCA` [5].

The second system employed was a locality-based version of `mg` in which term locality is used as a guide to relevance [1]. This run employed the *arc* shape formulation and the *logarithmic* height formulation.

A total of 86 retrieval runs were submitted (43 query sets * 2 retrieval runs). For both the document-based and locality-based versions of `mg`, the query variations `Sab3a`, `Sab1c` and `Sab1b` were most effective in terms of average precision, precision at 20 documents and reciprocal rank of first relevant document.

Acknowledgements

This work was supported by the Australian Research Council.

References

[1] O. de Kretser and A. Moffat. Effective document presentation with a locality-based similarity heuristic. In Marti Hearst, Fredric Gey, and Richard Tong, editors, *Proceedings of the 22nd International Conference on Research and Development in Information Retrieval*, pages 113–120, University of California, Berkeley, U.S.A., August 1999. ACM.

[2] Gerard Salton. *Automatic Text Processing*. Addison-Wesley, Reading, Massachusetts, 1989.

[3] A. Singhal, G. Salton, M. Mitra, and C. Buckley. Document length normalization. *Information Processing & Management*, 32(5):619–633, 1996.

[4] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and indexing documents and images.* Van Nostrand Reinhold, New York, 1994.

[5] Justin Zobel and Alistair Moffat. Exploring the similarity space. *ACM SIGIR Forum*, 32(1):18–34, Spring 1998.

Overview of the TREC-9 Web Track

David Hawking*
CSIRO Mathematical and Information Sciences,
Canberra, Australia
David.Hawking@cmis.csiro.au

September 4, 2001

Abstract

TREC-9 marked a broadening of the range of of search task types represented in the Web track and a serious attempt to determine whether hyperlinks could be used to improve retrieval effectiveness on a topic-relevance ad hoc retrieval task. The Large Web Task compared the ability of systems to locate online service pages within the 18.5 million page VLC2 collection. In this case the question is not whether the page is relevant to the topic, but whether it provides direct access to the desired service. In contrast, the Main Web Task compared link-based and non-link methods on a task involving topic relevance queries and a 1.69 million page corpus (WT10g) which was carefully engineered to ensure a high density of inter-server links and (relative) ease of processing. The Main Web task topics were in TREC Ad Hoc form but were reverse engineered from query logs. Ternary relevance judgments were obtained and, in addition, assessors were asked to identify "best" documents for each topic. As in TREC-8, no significant benefit associated with the use of link information in a topic-relevance retrieval task was demonstrated by any of the participating groups, whether or not additional weight was given to highly relevant documents.

1 Introduction

The TREC-9 Web Track activities centred on two tasks: the Main Task and the Large Task. The latter made use of the 100 gigabyte, 18.5 million webpage VLC2 collection described in the 1998 VLC Track overview [Hawking et al. 1998]. The former worked with a 10 gigabyte, 1.69 million document subset of the VLC2, distributed on five CD-ROMs as the WT10g collection. [Bailey et al. 2001].

A final Web Track activity was the invited talk to a TREC-9 plenary session by Dr Andrei Broder, Chief Scientist at the Alta Vista search engine company. Slides from an updated version of that talk, presented at the 2001 Search Engines Meeting, by Andrei's colleague Bob Travis, are available online [Travis and Broder 2001].

2 Main Web Task

None of the participants in the TREC-8 Small Web Task, using a two gigabyte corpus (WT2g), managed to demonstrate any benefit whatever from using hyperlink methods in that particular retrieval task. Given that most commercial Web search engines exploit hyperlinks to apparently great effect, this result may seem surprising.

Accordingly, a new task was devised for TREC-9 which removed possible impediments to good link-based performance which were perceived to be inherent in last year's task.

---

*The author wishes to acknowledge that this work was carried out partly within the Cooperative Research Centre for Advanced Computational Systems established under the Australian Government's Cooperative Research Centres Program.

2.0.1 Main Web: Task Summary

Corpus: A new test corpus was deliberately constructed in such a way as to dramatically increase the density of inter-server hyperlinks. Full details of the construction of the new corpus (known as WT10g) are documented elsewhere [Bailey et al. 2001]. WT10g was defined and initially distributed by the ACSys Cooperative Research Centre. Following the demise of ACSys in September 2000, WT10g has been distributed by CSIRO[1], which was one of the ACSys partners.

Note that, although WT10g is a subset of VLC2, documents in WT10g are assigned different document numbers to enable easy extraction of the document. However, they include the original document numbers within `<DOCOLDNO> ... </DOCOLDNO>` tags.

Topics: Test topics were reverse engineered from queries selected from real Web search engine[2] logs, rather than being generated with respect to the Ad Hoc corpus. The topics were presented in traditional Ad Hoc form with title, description and narrative fields. The title field contained the unedited query from the original log. The description and narrative fields were a statement of a particular interpretation of the query to be used in judging. For example, the polysemous query `cats` might have been interpreted as `Who wrote and who acted in the musical "Cats"?`. Documents relating to other meanings such as bulldozers and domestic pets would be judged as irrelevant.

Misspellings were a feature of several queries chosen as topics. An example was the single word query `angioplast7`. Not surprisingly, there were no occurrences of this word in the collection. Successful processing of the title-only version of this topic thus required either spelling-correction, approximate matching, or n-gram methods.

Results required: Participants were required to return a top 1000 (or fewer) list of documents for each topic, ranked in order of decreasing estimated relevance to the topic.

Judments: Ternary (irrelevant/relevant/highly relevant) rather than binary judging was adopted. Judges were also asked to identify *best* documents from among the highly relevants. Two additional judges were later asked to examine the relevant and highly relevant documents for each topic and to pick what they considered to be the best one (or possibly more than one.)

Focus: The principal focus of the Main Web Task was to re-attempt last year's question, in more favourable circumstances:

- Can link information in Web data be used to obtain more effective search rankings on a topic-relevance Ad Hoc retrieval task than can be obtained using page content alone?

2.1 Types of Run

A strong distinction must be drawn between automatic, title-only (*short*) runs and the rest (*notshort*). In real Web search, search engines only have access to the query recorded in the title field. The underlying information need is known only to the searcher and not to the search engine. Thus, only the short runs are representative of real Web search.

Despite this, notshort runs were encouraged because they add value to the test collection by increasing the number of known relevant documents. They also give some idea of what level of performance may be possible on each task and allow groups to continue work on longer queries.

The notshort category includes: interactive manual, blind manual, and automatic runs which used any part of the topics other than the title.

2.2 Judging pools

The number of runs judged was 59, giving a maximum pool size of 5900 (per topic). The mean actual pool size was 1401, 23.8% of the maximum, while the mean number of relevant documents over the 50 topics was 52.34, or 3.7% of the number of documents judged.

---

[1] Commonwealth Scientific and Industrial Research Organisation, an Australian government agency. (`mailto://test_collections@act.cmis.csiro.au`)

[2] eXcite

53.3% of the relevant documents were returned by both automatic and manual runs. 10.4% of the relevants were found by manual runs only, and another 15.3% were found by notshort runs only. Thus, if all runs other than the short ones were excluded, one quarter of the relevant documents would have been lost. Automatic runs (only) contributed 68.5% of the pools, manual (only) 15.6% of the pools, and so 15.8% of the pools were contributed by runs of both types.

Twenty-three groups submitted at least one run to the main web track, but only 21 submitted in time for runs to be judged. All of the 21 groups retrieved at least eight relevant documents in the top 100 across all 50 topics that no other group retrieved (*unique relevants*). The group that found the most unique relevants was Illinois Institute of Technology, with 212 over all topics. 162 of the 212 came from their manual run. The group with the next highest number of unique relevants was Hummingbird with 57 (over all topics). Justsystems and CUNY each had 55, with no one else having more than 50.

A total of 105 runs were submitted. Of these, 78 were content-only and 27 were content-link. There were 12 manual runs, 50 automatic short runs, and 53 automatic notshort runs.

2.2.1 Pool completeness - 1

NIST recently conducted experiments to determine what the effect of pooling fewer documents from each run would have been. Two cases were compared with official pools were based on the top 100 documents: top 50, and top 75.

Here are the results of that investigation:

Pool	Average pool size	Average number relevant
Top 100	1401.44	52.34(100%)
Top 75	1077.12	46.52 (88.88%)
Top 50	743.20	39.64 (75.74%)

NIST have decided on this basis to continue basing topic relevance pools on the top 100 documents. The drop in number relevant for 75 is not serious, but the pool size isn't reduced enough to be worthwhile (and one topic that had only 3 relevant would have lost 1). The drop in number of relevants for the top 50 case was considered too severe.

Unfortunately, it is not possible to reliably quantify what the effect of increasing the number of documents in the pools would have been. The following table shows how the probability that a pooled document will be judged relevant depends upon the rank at which it was retrieved.

Rank range	docs judged relevant
1-50	5.33%
51-75	2.06%
76-100	1.70%

Based on the very crude approximation to this data shown in Figure 1, we estimate the probability of documents in the 101-200 range being judged relevant if they had been included in the pool as 0.97%, suggesting that doubling the depth of judging might have increased the number of relevant documents per topic by an average of about 13 (about 25%).

It therefore seems almost certain that there are unjudged relevants within the collection. However, averaged over 50 topics, these are unlikely to affect relative system rankings.

2.2.2 Pool completeness - 2

Another way to look at pool completeness is to see how much judged runs' evaluations differ when using qrels with and without that group's unique relevants. NIST ran this computation for the TREC-9 web track (using both levels of relevant as "relevant") and mean average precision. The results are encouraging.

The largest percentage difference in mean average precision (MAP) is 10.3, but that run had poor effectiveness (MAP of .0174 using original qrels) and the absolute difference was only .0018. The next highest percentage difference was 6.1, again for a poor run. The third highest percentage difference was

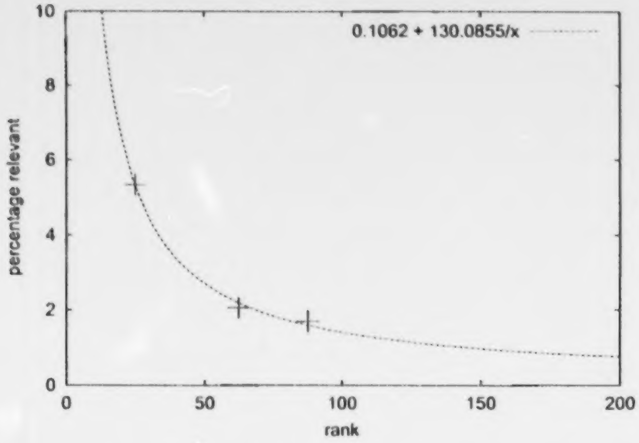


Figure 1: Decline in probability of document being judged relevant with increasing rank at which it was retrieved, shown with the $y = b + c/x$ line of best fit.

5.2 for the IIT manual run, the run which contributed the most unique relevants. For runs with a MAP of at least .1, the percentage difference was almost always less than 2%, except for some TNO runs that actually improved by around 3% when evaluated without their own unique relevants! When the process was repeated using P@10 instead of MAP, 38 of the 59 runs showed no change whatsoever. The biggest percentage difference was 7.72 for the IIT manual run.

2.2.3 Pool completeness - 3

How much of a problem is the presence of unjudged relevant documents in a test collection? Zobel [Zobel 1998] conducted various tests to try to determine how incomplete TREC collections relevance judgments are. He found that there were unjudged relevant, that the number of unjudged relevant was highly skewed by topic (the more relevant in early ranks, the more relevant there continues to be), and that the quality of the pools (diversity of systems contributing to the pools and depth in the system's ranking) did affect the quality of the resulting collection. But he also found that the TREC collections he looked at were quite acceptable for comparing retrieval systems – the errors he observed due to incompleteness were smaller than the differences occasioned by using different relevance assessors.

2.3 Properties of WT10g

In summary, WT10g is considerably larger than earlier TREC ad hoc collections and WT2g. However, ease of processing was improved by elimination of many of the binary and Non-English pages normally found in Web crawls. Most importantly, WT10g includes a very much higher density of inter-server hyperlinks than did WT2g. Readers are referred to [Bailey et al. 2001] for full WT10g collection properties.

Table 1 compares the densities of known relevant documents for the TREC-9 Main Web topics with that of other recent TREC collections. Naturally, there may be considerable variation from one topic to another.

2.3.1 Connectivity data

Nick Craswell's software for extracting hyper-link connectivity information from collections was run over WT10g and the resulting connectivity matrix was distributed with the collection on CD-ROM.

Table 1: The density of known relevant documents (across 50 topics) in WT10g for the TREC-9 topics compared to that in earlier tasks.

Judgments	Collection	Density of relevant docs
T7	VLC2	6482/18.5M = 0.03 %
T8	WT2g	2279/247491 = 0.92 %
T8	Ad Hoc	4728/528155 = 0.90 %
T9	WT10g	2371/1.69M = 0.14 %

2.4 Summary of participation

Tables 2 and 3 list the official runs submitted in the Short and Notshort categories. Runs which made use of link information are marked with the word LINK. Table 4 summarises the methods used by the Main Web participants.

2.4.1 Link v. Content

JustSystem Some pairs of runs showed an advantage arising from the use of anchor text on some measures. The biggest advantage was a few percent superiority for the `jscbt9wll2` run against the `jscbt9wcl1` baseline on all three measures. However, JustSystem conclude from a large set of unofficial as well as official runs that the benefit is small and inconsistent.

U Waterloo A minute gain on average precision was reported from the use of an unspecified link method for the T+D runs. In all other cases, use of links caused harm.

AT&T Use of relevance feedback `att0010gbt` harmed both early precision and average precision. The runs `att0010glf` and `att0010glv`, which used anchortext, performed worse on these measures than all the content-only runs except the relevance feedback run. Interestingly, there is no drop in performance for `att0010gbt` when runs are compared using DCG[100]. Upweighting query words in title fields (`att0010gbe`) was beneficial on all measures.

Other Runs Inspection of Tables 2 and 3 reveals no other indication of benefit achieved from use of link-based methods.

It is clear that this data and participant reports confirm last year's observation that no consistent benefit was gained from the use of links in this topic-relevance ad hoc retrieval task.

This seems to be also true, even when highly relevant documents are valued very highly.

2.4.2 Resilience to Query Misspelling

The effect of misspelled query words should be most noticeable in the short category. The five top-performing groups on that category used the following approaches:

JustSystem If insufficient results were returned in response to the original query, the query was automatically expanded to include spelling variations.

Hummingbird If an original query term occurred in fewer than ten documents, then Soundex-matching approximate match terms from the collection, with up to 2 edit errors were added. If the number of documents affected was still less than ten, then non-Soundex approximate matches were added. If necessary, trailing letters were dropped until the ten-document criterion was satisfied.

U Waterloo They found that 4-gram indexing worked better than words partly because of misspelling resilience but also because of conflation of morphological variants.

Table 2: All official runs submitted in the short (automatic, title-only) category, presented by group. Groups are ranked in order of decreasing average precision of their best run. The DCG[100] figures represent the discounted cumulative gain (described in the text) when a highly relevant document is considered to be worth 100 times as much as a relevant one. Natural logarithms were used ($b = e$) and the cutoff was at rank 100. The number of groups was 19 and the number of runs submitted was 40.

Group	Run tag	Ave. prec.	P@10	DCG[100]	Type
JustSystem	jscbt9wcs1	0.2011	0.238	107.024	
	jscbt9wls1	0.2000	0.252	110.027	LINK
	jscbt9wls2	0.1838	0.224	96.319	LINK
Hummingbird	hum9te	0.1970	0.254	118.725	
U Waterloo	uwmt9w10g0	0.1654	0.238	95.620	
	uwmt9w10g2	0.1631	0.236	95.356	LINK
	uwmt9w10g4	0.1812	0.240	94.366	
	uwmt9w10g5	0.1794	0.240	93.341	LINK
Twenty-One	tnout9t2	0.1801	0.214	107.745	
	tnout9t2lc10	0.1630	0.214	106.238	LINK
	tnout9t2lc50	0.1337	0.198	102.628	LINK
	tnout9t2lk50	0.0488	0.032	29.912	LINK
Ricoh	ric9tpx	0.1787	0.276	118.981	
U Neuchatel	NEtm	0.1754	0.212	104.484	
	NENRtm	0.1743	0.208	103.373	
	NENRtmLpas	0.1736	0.208	103.214	LINK
Queens CUNY	pir0Wt1	0.1750	0.218	99.975	
IIT/AAT/NCR	iit00t	0.1627	0.250	109.718	
ATT	att0010gb	0.1341	0.200	89.104	
	att0010gbe	0.1464	0.226	101.331	
	att0010gbl	0.1380	0.204	85.803	
	att0010gbt	0.1182	0.172	89.870	
	att0010glf	0.1250	0.182	89.579	LINK
	att0010glv	0.1288	0.190	90.983	LINK
Fujitsu Labs	Flab9atN	0.1360	0.194	102.861	
JHU/APL	apl9lt	0.1062	0.116	64.291	LINK
	apl9t	0.1272	0.134	77.119	
SABIR Research	Sab9web1	0.1265	0.182	98.120	
IRIT	Mer9Wt0	0.0996	0.128	56.285	
	MEr9Wt1	0.0114	0.070	14.546	
Seoul National U	Scai9web3	0.0915	0.154	83.513	
U Padova	PuShortAuth	0.0591	0.136	72.266	LINK
	PuShortBase	0.0654	0.142	74.769	
	PuShortWAuth	0.0637	0.138	73.591	LINK
UNC	isnnwt	0.0225	0.058	28.626	
	iswt	0.0240	0.076	38.807	
Pam Wood	UCCS1	0.0181	0.052	15.037	
	UCCS2	0.0169	0.052	16.792	
CWI	cwi0000	0.0176	0.066	13.609	
	cwi0010	0.0125	0.024	9.428	

Table 3: Measures as for Table 2. All official runs submitted in the notshort (manual, interactive, or non-title-only automatic) category, presented by group. Groups are ranked in order of decreasing average precision of their best run. The number of groups was 22 and the number of runs submitted was 65.

Group	Run tag	Ave. prec.	P@10	DCG[100]	Type
IIT/AAT/NCR	iit00td	0.2293	0.350	143.533	TD
	iit00tde	0.2217	0.346	143.306	TD
	iit00m	0.3519	0.518	172.199	M
JustSystem	jscbt9wcl1	0.2687	0.342	135.893	TDN
	jscbt9wll1	0.2659	0.344	132.645	TDN/LINK
	jscbt9wll2	0.2801	0.358	144.059	TDN/LINK
Ricoh	ric9dpn	0.2616	0.338	151.226	TD
	ric9dpx	0.2267	0.322	136.544	TD
	ric9dsx	0.2201	0.324	137.191	TD
	ric9dpxL	0.2257	0.316	135.891	M
U Neuchatel	NEnm	0.2499	0.342	142.946	TDN
	NEnmLpas	0.2488	0.340	142.417	TDN/LINK
	NEnmLsa	0.2185	0.332	136.446	TDN/LINK
ANU/CSIRO	acsys9mw0	0.2486	0.384	144.506	M
Hummingbird	hum9td4	0.2115	0.308	127.553	TD
	hum9tde	0.2217	0.294	121.085	TD
	hum9tdn	0.2335	0.352	139.349	TDN
Queens CUNY	pir0Wtd2	0.2164	0.302	122.122	TD
	pir0Wttd	0.2097	0.318	92.404	TD
	pir0WTTD	0.1418	0.180	92.404	TD/LINK
	pir0Watd	0.2209	0.298	130.067	TDN
Twenty-One	tnout9f1	0.2178	0.290	132.224	TDN
NeurOK	NRKlm	0.2064	0.282	126.202	TDN
	NRKprf20	0.2173	0.326	131.539	TDN
	NRKse10	0.1960	0.272	117.767	TDN
	NRKse20	0.1642	0.234	108.058	TDN
SABIR Research	Sab9web2	0.2122	0.340	134.015	TDN
	Sab9web3	0.2159	0.346	135.596	TDN
	Sab9web4	0.2091	0.342	134.333	TDN
	Sab9web5	0.2018	0.314	126.863	TDN/LINK
JHU/APL	apl9td	0.1917	0.340	120.747	TD
	apl9all	0.1948	0.314	125.502	TDN
	apl9ltdn	0.1494	0.232	98.921	TDN/LINK
	apl9tdn	0.1785	0.286	115.197	TDN
Fujitsu Labs	Flab9atd2N	0.1877	0.302	132.741	TD
	Flab9atdN	0.1816	0.298	139.320	TD
	Flab9atdnN	0.1923	0.316	139.320	TDN
SUNY Buffalo	xvsmmain	0.1521	0.214	110.224	TD
	xvsmtitle	0.1278	0.184	92.268	TN
	xvsmtdn	0.1694	0.238	113.566	TDN
	xvsmman	0.1785	0.260	119.496	M
Dublin City U	dcu00ca	0.1519	0.278	117.162	M
	dcu00la	0.1450	0.274	111.980	M/LINK
	dcu00lb	0.1324	0.244	102.915	M/LINK
	dcu00lc	0.1387	0.258	107.824	M/LINK
U Waterloo	uwmt9w10g1	0.1331	0.260	87.252	TD
	uwmt9w10g3	0.1336	0.262	87.860	TD/LINK
Seoul National U	Scai9Web1	0.0941	0.152	84.387	TD
	Scai9Web2	0.0934	0.138	82.733	TD
	Scai9Web4	0.0946	0.146	84.214	TD
RMIT/CSIRO	rmitNFGweb	0.0707	0.088	56.950	M
	rmitNFLweb	0.0702	0.090	54.944	M
	rmitWFGweb	0.0040	0.010	11.663	M
	rmitWFLweb	0.0341	0.044	27.193	M
U Padova	PuLongAuth	0.0648	0.172	81.156	TD/LINK
	PuLongBase	0.0666	0.180	83.536	TD
	PuLongWauth	0.0660	0.178	83.144	TD/LINK
UNC	iswtd	0.0325	0.110	42.449	TD
	iswtdn	0.0412	0.084	32.362	TDN
CWI	CWI0001	0.0174	0.054	14.872	TD
	CWI0002	0.0122	0.038	8.110	TDN
IRIT	Mer9WtdMr	0.0154	0.090	15.548	TD
	Mer9Wtnd	0.0140	0.092	14.758	TDN
Pam Wood	UCCS3	0.0000	0.000	0.000	D
	UCCS4	0.0000	0.000	0.000	D

BEST COPY AVAILABLE

Twenty-One Fuzzy matching?

Ricoh No reported correction.

2.4.3 Value of Query Expansion

Again considering only the short runs, the five top-performing groups reported the following:

JustSystem Found consistent improvement in average precision using both reference database feedback and pseudo-relevance feedback on WT10g. The combination achieved a gain of 16-17%.

Hummingbird An expansion method similar to Rocchio produced very small gain when evaluation was based on all relevant documents and caused harm when only highly relevants were considered.

U Waterloo No feedback employed.

Twenty-One Training with WT2g revealed that blind feedback caused harm due to large numbers of typographical errors in documents.

Ricoh In the official runs, query expansion caused harm. However, subsequent unofficial runs with a modified expansion method showed an improvement of 8% in average precision.

2.4.4 Short vs. Notshort

Table 5 compares the performance of the group of short runs versus the group of automatic notshort runs. As can be seen, there are substantial differences in favour of the notshort group. Comparing the medians for the groups, notshort is 49% better than short on P@10 and 44% better on average precision.

Best performance overall was achieved by the manual run `iit00m` from IIT/AAT/NCR. Its P@10 score was 88% better than the best P@10 score for a short run (`ric9tpx`, submitted by Ricoh). Its average precision score was 75% better than the best for a short run (`jscbt9wcs1`, submitted by JustSystem).

Note that the best possible P@10 score was 0.878, due to topics with less than 10 known relevant documents.

2.4.5 Evaluation by Highly Relevant Documents

Voorhees [Voorhees 2001] found that WT10g system rankings based on Highly Relevant judgments were non-trivially different from those based on Relevant and Highly Relevant combined. Because the numbers of Highly Relevant documents are relatively small she recommended the use of the ***Discounted Cumulative Gain*** (DCG) method proposed by Järvelin and Kekäläinen [Järvelin and Kekäläinen 2000] to combine information from both categories of relevance, but with higher weight to the Highly Relevants.

Tables 2 and 3 report DCG scores for the Main Task runs with a heavy bias toward Highly Relevant.

2.4.6 Evaluation by Best Documents

Voorhees [Voorhees 2001] found little agreement between judges about which pages were the best resources on a topic. She found that best page judgments did not lead to stable measures.

However, the best page judgments give the opportunity to look at whether the best resources on a topic tend to be: a) site homepages, b) close to the root of a directory tree, c) documents with higher than average in-link count. The following analysis uses the union of the sets of bestpage judgments for each of the three judges.

Site homepages: I examined each of the 130 documents identified by one or more of three judges as the best for a topic. I classified only one of them as a site home page (a small site published by the Rainbird Company about the Rose Parade). There were also three Yahoo! directory pages but most of the best documents were individual pages which presented detailed information on the topic.

Depth in directory hierarchy: On average, the best pages were 2.61 levels deep within the directory hierarchy, compared with an average of 2.96 levels for all pages in the collection.

Table 4: Style of link exploitation methods used by groups participating in the Main Web Task. In many cases, the methods actually employed represent modifications of the basic method listed.

Group	Methods
ATT	Anchor Text Propagation
Dublin City U	Inlink Count
Dublin City U	Spreading Activation
Dublin City U	HITS/Co-citation
JHU/APL	Inlink Count
JustSystem	Anchor Text Propagation
Queens College CUNY	Inlink Count
SABIR	Not Stated
Twenty-One	HITS
Twenty-One	Co-citation
U Neuchatel	Spreading Activation
U Neuchatel	Probabilistic Argumentation
U Neuchatel	HITS
U Neuchatel	PageRank
UPadova	HITS
UPadova	HITS/Similarity
U Waterloo	Not Stated

Table 5: Comparative performance of short v. notshort runs (excluding manual runs).

Measure	Short	Notshort
P@10 (best)	0.276	0.358
P@10 (median)	0.198	0.296
Ave Prec (best)	0.2011	0.2801
AvePrec (median)	0.1341	0.1936

Directory depth	1	2	3	4	5	6	7	8
No. of Bests	32	32	38	14	13	-	-	1

In-link counts: Based on the `in_links.gz` file distributed with WT10g, there are 8,062,918 inlinks across the 1,692,096 documents in the collection, giving an average number of inlinks per page of 4.77. Nick Craswell has computed inlink counts for each of the judging categories.

Category	All docs	Docs Judged Irrelevant	Docs Judged Relevant	Docs Judged Highly Rel.	Bests
Mean Inlinks/page	4.77	8.48	5.26	4.80	4.67
Median Inlinks/page	1	1	1	1	2

The inlink distribution for the collection is very heavily skewed, with a high peak at one. The fact that documents judged to be irrelevant show a much higher inlink density than the average for the collection, probably results from the unsuccessful use of link-based methods by participants. It is quite interesting that relevant and highly relevant documents are not distinguished from randomly chosen documents by their inlink count. Indeed the fact that the median for both groups is 1 indicates that more than half of them have 1 or fewer inlinks.

Superficially, it may seem that there is an exploitable difference between the inlink counts for the best pages and for pages in general. The raw average inlink count for best pages is 9.78, however that figure is grossly distorted by an extreme outlier with 337 incoming links. That single page accounts for 53% of all incoming links to best pages. Excluding it reduces the average to 4.67, the figure reported in the table. In fact, the median score for the bests is barely above one.

It appears that best pages are not distinguishable in any useful way from pages in general or from less relevant pages on any of the three attributes considered.

2.5 Main Web Task discussion and conclusions

Evidence is already available [Bailey et al. 2001] that WT10g is in fact large enough and contains sufficient links to demonstrate a dramatic advantage to a link based method on a homepage finding task. It seems reasonable to conclude that link methods can be beneficial in some forms of retrieval task, even over a small test collection like WT10g.

However, no such advantages have been found for topic relevance tasks. Even though consideration of multiple degrees of relevance does change relative system rankings, it still does not allow demonstration of any worthwhile advantage to link methods. Indeed, even the best resources for a topic appear not to be usefully distinguishable by frequency of incoming links.

3 Appropriate Web Evaluation Methodology

Following robust and mutually beneficial debate at the Infonortics Search Engines 2000 meeting[3], the Web Track organisers became convinced of the need to significantly extend TREC Ad Hoc evaluation methodology to accommodate different types of retrieval task and to use appropriate judging instructions and measures for each type.

TREC Ad Hoc retrieval exercises (including previous VLC and Web tasks) have concentrated on topic relevance tasks in which a searcher is assumed to be looking for a range of documents which are relevant to a particular research topic. Such retrieval tasks are definitely represented in Web search engine query streams but form only a small proportion of the total.

Following the Search Engines 2000 meeting, CSIRO/ANU have proposed a taxonomy in which search tasks are classified at the top level by how many results the searcher expects:

---

[3] `http://www.infonortics.com/` The debate particularly involved David Hawking and Chris Buckley representing TREC evaluation and (among others) Larry Page of Google, Eric Brewer of Inktomi and Andrei Broder of Alta Vista, representing Search Engine companies. David Evans was the moderator.

- a part document (eg. Q&A),
- a single document (eg. known-item and homepage search),
- a selection of documents (This class may include topic relevance and online-service location tasks), and
- all documents matching a criterion (eg. metadata search, such as all documents authored by a particular person).

CSIRO/ANU have conducted evaluations of public search engines using on-line service location (selection of pages) and homepage finding (single page) [Hawking et al. 2001; Hawking et al. 2001; Craswell et al. 2001] as well as topic relevance tasks. On-line service evaluation was also used in the TREC-9 Large Web task.

In his TREC-9 talk, Andrei Broder presented an alternative taxonomy of Web search, dividing searches into:

- Informational,
- Transactional, and
- Navigational

classes, each of which require different evaluation methods.

The TREC-2001 Web Track includes a homepage finding task.

4 Large Web Task

In the past, participation in the VLC track and the Large Web track was limited by the difficulties of processing 100 gigabytes of data. This year however, the number of participants declined, due to the fact that 18.5 million pages (100 gigabytes) no longer constitutes an interesting challenge to those seriously pursuing scalable, large scale retrieval. At the time of the TREC-9, major Web search engines were indexing around 30 times as many pages as are contained in the VLC2 collection.

A number of different objectives were pursued by the individual participants. They are summarised as follows:

ACSys Comparing anchor text and PageRank resorting methods with Okapi BM25.

AT&T Testing the new retrieval system Tivra. Comparing anchor text with content only.

Fujitsu Labs Engineering to achieve a good balance between speed, effectiveness and cost.

Hummingbird Evaluating an experimental version of Fulcrum SearchServer on large data. Testing approximate search.

U Waterloo Repeat of TREC-8 runs.

4.1 Large Web Task: Topics and assessments

The 10,000 "natural language" queries from the TREC-8 Large Web task were re-used. They were obtained by random selection from combined large Alta Vista and Electric Monk search logs and were numbered 20001-30000.

Participants were required to process all 10000 queries and to submit top 10 rankings to ACSys for judging. After submissions were received, the track coordinator (David Hawking, who was not an official participant in the TREC-9 Large task) selected 106 of the topics which seemed to have been intended to locate some form of online service. Four of these were used as practice by the judges. Sample accepted queries are shown in Figure 2.

The pooled documents for each topic were presented to the assessors in order of increasing document length using the RAT (Relevance Assessment Tool [Hawking et al. 2001]) used in previous VLC track experiments. This time however, a text-only web browser [Lynx] was used to display documents in a way which rendered references and tables in a reasonable way (minus images).

The four assessors were all University graduates from specialties other than Computer Science or Librarianship. Two of them had served as VLC track judges in previous years.

During judging zero "good" documents were found for 18 queries, which are therefore excluded from the following analysis.

The number of good documents found per query ranged from 1 to 72, with a mean of 24.1. The number of queries for which fewer than five good documents were found was 14 and 21 had fewer than 10. A total of 6911 documents were judged.

4.2 Runs

All runs judged in the Large Task are listed in Table 6. The P@10 results are also shown graphically in Figure 3. Figure 4 shows the tradeoffs between cost, speed, space and effectiveness for the runs for which detailed information was provided.

The runs in Table 6 which are labelled with an asterisk were "submitted" by Nick Craswell who was a PhD student in ACSys (to June 2000), then an intern at Microsoft Research, Cambridge (June-September 2000) and subsequently an employee of CSIRO. The `acsys9*` runs constituted an experiment in use of hyperlink methods and were in fact judged blindly but cannot be fairly compared with those of other participants.

Nick used the TREC-8 Microsoft Research run as a baseline. This was possible because TREC-8 Large Task participants had submitted results for all 10,000 queries, not just the 50 or so which were judged in TREC-8. The run `acsys9pr` was a PADRE run in which the top 1000 documents were reranked on the basis of PageRank scores computed for the VLC2 collection. The other two ACSys runs made use of link anchor text. All links whose anchor text matched the query were included and documents were scored on the basis of a count of incoming matching links. Run `acsys9lnkA` included all matching links whereas `acsys9lnkE` excluded within-site links.

4.2.1 Large Web Notes

Hardware/OS AT&T, Fujitsu and Hummingbird all used single low-cost PC systems. AT&T and Fujitsu used Linux and Hummingbird used Windows NT.

Cost Fujitsu set a new mark for the cheapest system used to run the Large Web task. With a $US1700 dual-Celeron system (648 MB RAM, and 3 x 40gB disks) they indexed the data in just over 12 hours (including decompression) and were able to process queries in an average of 0.31 sec.

Comparison with TREC-8 The Microsoft run evaluated in both TREC-8 and TREC-9 achieved a P@10 of around 0.56 in TREC-8 and a slightly lower figure of 0.52 in TREC-9. By contrast, the University of Waterloo TREC-8 runs achieved 0.58-0.59 but identical runs in TREC-9 came in at only 0.43-0.45. The differences between Waterloo and Microsoft on TREC-9 have not been subjected to statistical significance testing but if the differences were significant it would be interesting to investigate whether the algorithms used by Waterloo worked better on a topic relevance rather than an online-service finding task.

Link methods AT&T were surprised to find no benefit from use of anchor text in this task. ACSys also found no benefit from anchor text or PageRank reranking relative to the Microsoft baseline, but the scoring methods used in the anchor text case were not very sophisticated.

5 Conclusions

No conclusive or consistent benefit from the use of link information was demonstrated on the Main Task, despite a larger (10 gB) dataset, a relatively high density of inter-server links and the use of assessments

```
25209 where can i find love songs?
25363 where can i find cd rom drivers
25418 any packaging business for sale?
25538 where can i find some frank frazetta wallpaper?
25744 mp3
25819 where can i shop for toys?
25861 where can i learn dutch
25989 where can i find mortgage rates
26070 precision engineering instruments
26075 where can i find html editors?
26161 where can i find an online translator?
26360 where can i download winnie the pooh and tigger
26465 how do i find someone's phone number
26487 how can i send flowers?
```

Figure 2: A sample of the judged queries used in the Large Web task.

Table 6: All official runs submitted in the Large Web task, presented by group. Groups are ranked in order of decreasing average P@10 of their best run. The number of groups was 6 and the number of runs "submitted" was 15. Unofficial runs inserted by the organisers (see text) are marked with an asterisk.

Group	Run tag	P@1	P@5	P@10
Microsoft*	ok8v1	0.5000	0.5214	0.5083
AT&T	att00100gb	0.5357	0.5190	0.4964
AT&T	att00100glf	0.5476	0.5048	0.4738
ACSys*	acsys9lnkA	0.4524	0.4643	0.4583
ACSys*	acsys9lnkE	0.5119	0.4929	0.4905
ACSys*	acsys9pr	0.3452	0.3167	0.2762
U Waterloo	uwmt9w100g0	0.4881	0.4500	0.4536
U Waterloo	uwmt9w100g1	0.4167	0.4262	0.4274
U Waterloo	uwmt9w100g2	0.4643	0.4548	0.4274
Fujitsu Labs	Flab9bN	0.4405	0.4619	0.4452
Fujitsu Labs	Flab9bsN	0.4524	0.4595	0.4381
Fujitsu Labs	Flab9rN	0.4405	0.4548	0.4440
Hummingbird	hum9w1	0.3095	0.3262	0.3250
Hummingbird	hum9w2	0.3095	0.3143	0.3167
Hummingbird	hum9w3	0.2857	0.3024	0.2940

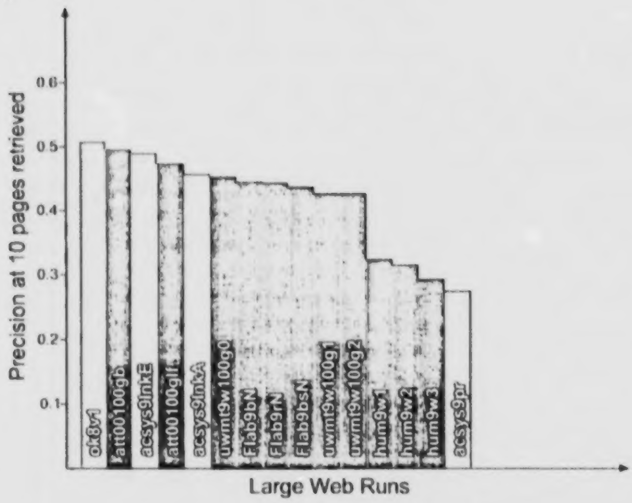


Figure 3: P@10 results for Large Task runs. Runs corresponding to the lighter coloured (orange) bars were submitted by the organisers and should not be compared with the other runs.

based on multiple levels of relevance, including identification of best pages. This finding is specific to a topic relevance task and is considered unlikely to apply to other forms of search task.

In the online service location task using the 100 gigabyte VLC2 collection, use of anchor text enabled AT&T to retrieve one more good document at rank one but otherwise no benefit was demonstrated on this task either. However, further work on this task is needed as the number of runs was small and there was little opportunity for tuning.

Useful information regarding the differences between Web and other TREC data has been accumulated through this year's Web track. Hopefully this will lead to increased participation in TREC-2001 and the use of better tested code.

Acknowledgements

With assistance from her colleagues at NIST, Ellen Voorhees played a major role in organising the Main Web part of the track, through topic formulation, assessment, evaluation and analysis. Much of the Main Web data and many of the analyses reported here are the result of her work.

The pivotal contributions of Peter Bailey and Nick Craswell in engineering the WT10g and preparing connectivity and other data are gratefully acknowledged.

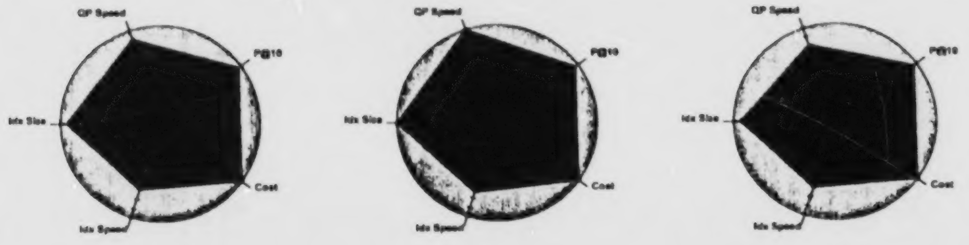
We are very much indebted to Brewster Kahle of the Internet Archive for making available the spidered data from which the VLC2 and WT10g collections are derived and to Alta Vista (Monika Henzinger and Michael Moricz), eXcite (Jack Xu) and the Electric Monk (Edwin Cooper) for providing large samples of queries from their logs. Thanks also to John O'Callaghan and Darrell Williamson (successive CEOs of ACSys) and Peter Langford (ACSys Centre Manager) for supporting the track.

Finally, thanks are due to the NIST assessors for the Main Task assessments and to Sonya Welykyj, Penny Craswell, Julie Lemon, and Andrew Duncan for their work in assessing Large Task submissions.

Bibliography

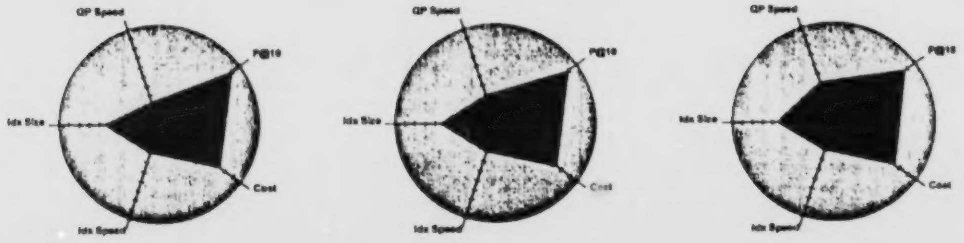
Bailey, P., Craswell, N., and Hawking, D. 2001. Engineering a multi-purpose test collection for

Fujitsu Laboratories



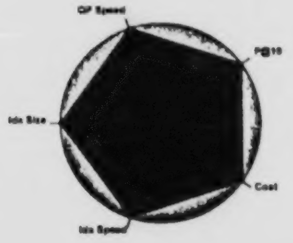
Flab9bN Flab9baN Flab9rN

Hummingbird



hum9w1 hum9w2 hum9w3

An Ideal Case



The All-Round Best System

TREC-9 Large Web Submissions - Log Scaling

Figure 4: The trade-off between cost, speed, space and effectiveness for the runs for which questionnaire responses were received. Index size, indexing time and query processing times are scaled relative to the best values known to have been achieved in either TREC-8 or TREC-9.

BEST COPY AVAILABLE

web retrieval experiments. *Information Processing and Management*. In press. `www.ted.cmis.csiro.au/~dave/cwc.ps.gz`.

CRASWELL, N., HAWKING, D., AND GRIFFITHS, K. 2001. Which search engine is best at finding airline site home pages? Technical Report XXX, CSIRO Mathematical and Information Sciences. `www.ted.cmis.csiro.au/~nickc/pubs/airlines.pdf`.

HAWKING, D., CRASWELL, N., BAILEY, P., AND GRIFFITHS, K. 2001. Measuring search engine quality. *Information Retrieval 4*, 1, 33–59.

HAWKING, D., CRASWELL, N., AND GRIFFITHS, K. 2001. Which search engine is best at finding online services? In *WWW10 Poster Proceedings* (May 2001). `www.ted.cmis.csiro.au/~dave/www10poster.pdf`.

HAWKING, D., CRASWELL, N., AND THISTLEWAITE, P. 1998. Overview of TREC-7 Very Large Collection Track. In E. M. VOORHEES AND D. K. HARMAN Eds., *Proceedings of TREC-7* (Gaithersburg MD, November 1998), pp. 91–104. `http://trec.nist.gov/pubs/trec7/t7\_proceedings.html`.

JÄRVELIN, K. AND KEKÄLÄINEN, J. 2000. Ir methods for retrieving highly relevant documents. In *Proceedings of SIGIR'00* (Athens, Greece, 2000), pp. 41–48.

LYNX. Lynx browser home page. `http://lynx.browser.org`.

TRAVIS, B. AND BRODER, A. 2001. Web search quality vs. informational relevance. In *Proceedings of the 2001 Infonortics Search Engines Meeting* (Boston, 2001). `www.infonortics.com/searchengines/sh01/slides-01/travis.html`.

VOORHEES, E. 2001. Evaluation by highly relevant documents. In *Proceedings of SIGIR'01* (New Orleans, LA, 2001). To Appear.

ZOBEL, J. 1998. How reliable are the results of large-scale information retrieval experiments? In *Proceedings of ACM SIGIR'98* (Melbourne, Australia, August 1998).

BEST COPY AVAILABLE

The Mirror DBMS at TREC-9

Arjen P. de Vries
arjen@acm.org
CWI, Amsterdam, The Netherlands

Abstract

The Mirror DBMS is a prototype database system especially designed for multimedia and web retrieval. From a database perspective, this year's purpose has been to check whether we can get sufficient efficiency on the larger data set used in TREC-9. From an IR perspective, the experiments are limited to rather primitive web-retrieval, teaching us that web-retrieval is (un?-)fortunately not just retrieving text from a different data source. We report on some limited (and disappointing) experiments in an attempt to benefit from the manually assigned data in the metatags. We further discuss observations with respect to the effectiveness of title-only topics.

1 Introduction

The Mirror DBMS [dV99] combines content management and data management in a single system. The main advantage of such integration is the facility to combine IR with traditional data retrieval. Furthermore, IR researchers can experiment more easily with new retrieval models, using and combining various sources of information. The IR retrieval model is completely integrated in the Mirror DBMS database architecture, emphasizing efficient set-oriented query processing. The logical layer of its architecture supports a nested object algebra called Moa; the physical layer uses the Monet main-memory DBMS and its MIL query language [BK99]. Experiments performed in last year's evaluation are described in [dVH99]; its support for IR is presented in detail in [dV98] and [dVW99].

The main goal of this year's participation in TREC has been to migrate from plain text retrieval to retrieving web documents, and simultaneously improve our algorithms to handle the significantly larger collections. The paper is organized as follows. Section 2 details our lab environment. Section 3 interprets our results and discusses our plans for next year with the Mirror DBMS, followed by conclusions.

BEST COPY AVAILABLE

```
<doc docno='WTX044-B44-57' doceldno='IA044-000798-B032-287'>
<title>bootnet reviews</title>
<description> ... </description>
<keywords> ... </keywords>
<body>
compaq ambitious delivers impressive design performance and features compaq presario so
near yet so far win means the issue of sound blaster compatibility is dead and buried right
wrong ...
</body>
<img>card cage</img>
</doc>
```

Figure 1: The intermediate format produced (XML).

2 Lab Environment

This section discusses the processing of the WT10g data collection used to produce our runs. The hardware platform running the experiments is a (dedicated) dual Pentium III 600 MHz, running Linux, with 1 Gb of main memory and 100 Gb of disk space.

Adapting our existing IR setup to handling Web data caused much more trouble than expected. As a side-effect of this problem, the submitted runs contained some errors, and even fixing does not give us the best runs ever; a lot of work still remains to be done to improve our current platform.

Managing a new document collection and getting it indexed has turned out to be a rather timeconsuming problem. Obviously, the WT10g is 10 times larger than TREC, so we decided to treat it as a collection of 104 subcollections following the layout on the compact discs. But, handling a collection *of this size* was not the real issue; our main problems related to the 'quality' of data gathered from the web.

2.1 Parsing

After some initial naive efforts to hack a home-grown HTML parser, we bailed out and used the readily available Perl package `HTML::Parser`. It is pretty good at 'correcting' bad HTML on-the-fly; the only real problem we bumped into was that it assumes a document to always have at least a `<HEAD>` and a `<BODY>`, which fails on `WTX089-B33`.

We convert the sloppy HTML documents into (rather simple) XML documents that are easier to manage in the subsequent indexing steps. We keep the 'normal' content words, the content of `<IMG>`'s `ALT` attribute, as well as the following meta tags: **`keywords`**, **`description`**, **`classification`**, **`abstract`**, **`author`**, **`build`**. In this first step, we also normalize the textual data to some extent by converting to lower-case and throwing out 'strange characters'; unfortunately, due to working against a very tight schedule (*too* tight), this included the removal of all punctuation and numeric characters (not helping topics referring to a particular year, like topics 456 and 481). An example result file is shown in Figure 1.

What affected our results severely is our assumption that HTML documents are nicely wrapped in `<HTML>` and `</HTML>` tags. Unfortunately, this first 'bug' removed about

BEST COPY AVAILABLE

half of the collection from our index.

2.2 Indexing

The second step reads the intermediate XML files and converts them into load tables for our database system. The contents of the `title`, `body`, and `img` tags are unioned together in 'term' sets, and all other tags in 'meta' sets.[1] Notice that we do not *have* to union these tags together; but, any alternative requires an IR model that can handle fields properly, which is still beyond our skill.

After loading these tables, the complete collection is represented by the following schema:

```
define WT10g_docs as
SET<
  SET<
    TUPLE<
      Atomic<string>         : docno,
      SET< Atomic<string> > : term,
      SET< Atomic<string> > : meta
    >
  >: subCollection
>;
```

The Mirror DBMS supports efficient ranking using the CONTREP *domain-specific* Moa *structures*, specialized in IR. These structures now have to be created from the above representation; this indexing procedure is still implemented in a separate indexing MIL script which is directly fed into Monet, the DBMS. The script performs stopping, stemming, creates the global statistics, and creates the internally used $< d_j,\ t_i,\ tf_{i,j} >$-tuples.

Web-data is quite different from newspaper articles: the strangest terms can be found in the indexing vocabulary after stopping and stemming. Examples vary from 'yippieyayeheee' and the like, to complete sentences lacking spaces between the words. After quick inspection, we decided to prune the vocabulary aggressively: all words longer than 20 characters are plainly removed from the indexing vocabulary, as well as all words containing a sequence of more than four identical characters.[2]

2.3 Retrieval

After running the indexing script, we obtain the following schema that can be used in Moa expressions to perform ranking:

---

[1] Notice that these 'sets' are really *multi-sets* or *bags*.

[2] We realize that this is a rather drastic ad-hoc approach; it is not likely to survive into the codebase of next year.

```
define WT10g_index as
SET<
  SET<
    TUPLE<
      Atomic<string>: docno,
      CONTREP       : term,
      CONTREP       : meta
    >
  >: subCollection
>;
```

The Monet database containing both the parsed web documents and its index takes 9 Gb of disk space.

Like in TREC-8, we use Hiemstra's LMM retrieval model (see also [Hie00] and our technical report [HdV00]). It builds a simple statistical language model for each document in the collection. The probability that a query $T_1, T_2, \cdots, T_n$ of length n is generated by the language model of the document with identifier D is defined by the following equation:

$$P(T_1 = t_1, \cdots, T_n = t_n | D = d) = \prod_{i=1}^{n} (\alpha_1 \frac{df(t_i)}{\sum_t df(t)} + \alpha_2 \frac{tf(t_i, d)}{\sum_t tf(t, d)}) \qquad (1)$$

The `getBL`-operator (i.e. get beliefs) defined for the CONTREP structure ranks documents according to this retrieval model. We planned two series of experiments: ranking using the raw content collected in the 'term' sets, and ranking using a weighted combination from the first ranking with the ranking based on the annotations extracted from the meta tags collected in the 'meta' sets. These two experiments are (approximately) described by the following Moa expressions:[3]

```
(1) flatten(map[map[TUPLE<THIS.docno,
      sum(getBL(THIS.term, termstat, query))>](THIS)](WT10g_index));

(2) map[TUPLE< THIS.docno, THIS.termBel + 0.1*THIS.metaBel >](
    flatten(map[
        map[ TUPLE< THIS.docno,
          sum(getBL(THIS.term, termstat, query)): termBel,
          sum(getBL(THIS.meta, metastat, query)): metaBel >
        ](THIS)
      ]( WT10g_index )));
```

We do not expect the reader to grasp the full meaning of these queries, but only intend to give an overall impression; the inner `map` computes the conditional probabilities for documents in a subcollection, which are accessed with the outer `map`; the `flattens` remove nesting. Similar to TREC-8, the query plan generated by the Moa rewriter (in MIL) has been manually edited to loop over the 50 topics, log the computed ranking for each topic, and use two additional tables, one with precomputed normalized inverse document frequencies (a materialized view), and one with the document-specific constants for normalizing the term frequencies. The `flatten` and the outer `map`, which iterate over the 104 subcollections and merge the results, were written directly in MIL as well. Unfortunately, we introduced a second (real) bug in this merging phase, which messed up our main results: we used `slice(1,1000)` whereas this really selects from the 2nd up to the 1001st ranking document; hence throwing out the 104 'best' documents (best according to our model).

---

[3] Details like sorting and selecting the top ranked documents have been left out.

3 Discussion

Table 1 presents a summary of the average precision (AP, as reported by `trec_eval`) measured on our runs. The first column has the results in which the top 100 documents are missing; the second column has the fixed results.[4]

run name	description	AP	AP (fixed)
CWI0000	title	0.0176	0.1814
CWI0001	title & description	0.0174	0.1503
CWI0002	title & description & narrative	0.0122	0.1081
CWI0010	title (term & meta)	0.0125	–

Table 1: Result summary.

Very surprising is the fact that using the description and/or narrative has not been helpful at all. This is completely different from our experience with evaluating TREC-6 and TREC-8 topics. Closer examination shows that the description (and sometimes the narrative) help significantly for the following topics:

- 452: 'do beavers live in salt water?'. Here, the description adds more general words such as 'habitat';
- 455: the description specifies 'major league' and the narrative gives finally the desired 'baseball';
- 476: the description adds the scope ('television' and 'movies') to title 'Jennifer Aniston';
- 478: The description adds 'mayor' to 'Baltimore'

---

[4] The run with combined term and meta data has not been fixed, due to some strange software problems that we have not figured out *yet*.

- 498: The title does not mention 'how many' and 'cost' which reveal the real information need.

Precision drops however in most other cases; especially the very concise title queries 485 ('gps clock'), 497 ('orchid') and 499 ('pool cue') suffer from overspecification in description and narrative. For example, topic 486 shows that the 'casino' from the title is not weighted enough in comparison to generic terms like 'Eldorado'. It warrants further investigation to view if query term weighting would address this counter-intuitive result.

We have not succeeded to make effective use of the information collected in the meta tags. Using *only* the meta tags leads to a poor average precision of 0.0033. Closer investigation of topics 451, 492, 494, for which the meta tags do retrieve relevant documents in the top 10, it turns out that these are also ranked high using the bare document content. Thus, we can only conclude that in our current approach, the meta tags can safely be ignored. Despite of this disappointing experience, we hope still that using this information source may be more benificial for processing blind relevance feedback.

Table 2 shows the results after spell-checking the topics semi-automatically using a combination of `ispell` and common sense, showing that this would have a minor positive effect on the title-only queries. Results for topics 463 (Tartin), 475 (compostion), and 487 (angioplast7) improve significantly; but, splitting 'nativityscenes' in topic 464 causes a loss in precision, because the (Porter) stemmer reduces 'nativity' to 'nativ'. We conclude from the other runs with spelled topics that we should address proper weighting of title terms first.

run name	description	AP
CWI0000s	title	0.1924
CWI0001s	title & description	0.1525
CWI0002s	title & description & narrative	0.1085

Table 2: Results with spell-checked topics.

4 Conclusions

The honest conclusion of this year's evaluation should be that we underestimated the problem of handling Web data. Surprising is the performance of the title-only queries doing better than queries including description or even narrative. It seems that the web-track topics are really different from the previous TREC topics in the ad-hoc task, for which we never weighted title terms different from description or narrative.

For next year, our primary goal will be to improve the current indexing situation. The indexing process can be described declaratively using the notion of *feature grammars* described in [dVWAK00]. Also, we will split the indexing vocabulary in a 'trusted' vocabulary (based on a proper dictionary), a numeric and named entity collection, and a 'trash' dictionary with rare and possibly non-sense terms. A third goal should be evident from the following quote from last year's TREC paper:

> Next year, the Mirror DBMS should be ready to participate in the large WEB track.

In other words: we still intend to tackle the large web track. For our cross-lingual work in CLEF [dV00] we have to address the problems of spelling errors and named entities anyways, which is directly related to some of our TREC problems. Some other ideas include to work on using links, blind relevance feedback, as well as improve our understanding on how to effectively exploit the 'meta' sets.

Acknowledgements

Henk Ernst Blok did a great just-in-time job of getting a late delivery of hardware prepared for running these experiments. Further thanks go to Djoerd Hiemstra and Niels Nes for their support with IR models and Monet respectively.

References

[BK99] P.A. Boncz and M.L. Kersten. MIL primitives for querying a fragmented world. *The VLDB Journal*, 8(2):101–119, 1999.

[dV98] A.P. de Vries. Mirror: Multimedia query processing in extensible databases. In *Proceedings of the fourteenth Twente workshop on language technology (TWLT14): Language Technology in Multimedia Information Retrieval*, pages 37–48, Enschede, The Netherlands, December 1998.

[dV99] A.P. de Vries. *Content and multimedia database management systems*. PhD thesis, University of Twente, Enschede, The Netherlands, December 1999.

[dV00] A.P. de Vries. A poor man's approach to CLEF. In *CLEF 2000: Workshop on cross-language information retrieval and evaluation*, Lisbon, Portugal, September 2000. Working Notes.

[dVH99] A.P. de Vries and D. Hiemstra. The Mirror DBMS at TREC-8. In *Proceedings of the Eighth Text Retrieval Conference TREC-8*, number 500-246 in NIST Special publications, pages 725–734, Gaithersburg, Maryland, November 1999.

[dVW99] A.P. de Vries and A.N. Wilschut. On the integration of IR and databases. In *Database issues in multimedia; short paper proceedings, international conference on database semantics (DS-8)*, pages 16–31, Rotorua, New Zealand, January 1999.

[dVWAK00] A.P. de Vries, M. Windhouwer, P.M.G. Apers, and M. Kersten. Information access in multimedia databases based on feature models. *New Generation Computing*, 18(4):323–339, August 2000.

[HdV00] Djoerd Hiemstra and Arjen de Vries. Relating the new language models of information retrieval to the traditional retrieval models. Technical Report TR-CTIT-00-09, Centre for Telematics and Information Technology, May 2000.

[Hie00] D. Hiemstra. A probabilistic justification for using tfidf term weighting in information retrieval. *International Journal on Digital Libraries*, 3(2):131–139, 2000.

Dublin City University Experiments in Connectivity Analysis for TREC-9

Cathal Gurrin & Alan F. Smeaton

School of Computer Applications
Dublin City University
Ireland
cgurrin@compapp.dcu.ie

Abstract

Dublin City University (DCU) took part in the Web Track (small task) in TREC-9. Our experiments were based on evaluating a number of connectivity analysis algorithms that we hoped would produce a marked improvement over a baseline Vector Space model system. Our connectivity experiments are all based on non-iterative post-query algorithms, which rerank a set of documents returned from content-only VSM queries. We feel that in order to implement a real-world system based on connectivity analysis the algorithms must have a low query-time processing overhead, hence our employment of non-iterative algorithms. Our results showed that we were unable to improve over a content-only run with our algorithms. We believe this to be mainly due to the nature of the link structure within the WT10g dataset.

1. Introduction

Dublin City University (DCU) took part in the Web Track (small task) for TREC-9. We wished to continue the experiments we carried out last year on the WT2g TREC-8 dataset. For additional information on our experiments for TREC-8 please see [1]. Our experiments were based on evaluating a number of connectivity analysis algorithms, which we hoped would produce a marked improvement over a baseline Vector Space model system. Our connectivity experiments are all based on non-iterative post-query algorithms, which rerank a set of documents returned from the content-only VSM queries. We feel that in order to implement a real-world system based on connectivity analysis the algorithms must have a low query-time processing overhead, or all required processing must be done at indexing time.

We outline, in this document, details of our content based and linkage-based runs, which were aimed at ranking the most relevant and useful documents at the top of the search results. In effect we are attempting to improve the precision (over the baseline result) of the top documents returned from a search because the vast majority of users only look at the first page of search results. Recall that almost 85% of users only look at the top 10 results. Our approach is based on the assumption that, by implementing a conventional text-based search on the dataset, a subset of documents that are relevant to the topic in question will be generated. The execution of various linkage-based formulae on this small subset of documents will then increase the ranking of the most popular/best documents contained therein.

We do this by developing three algorithms for generating a connectivity score (Sc'_n) for each document in a set of relevant documents. In so doing we must distinguish between the two semantically different types of links to be found on the WWW of today, discarding the less useful types from our processing. This paper assumes certain knowledge about connectivity analysis. For those requiring an introduction we recommend our own TREC-8 article, see [1] or Li's description of the Hyperlink Vector Voting method [2] which ranks a document on the basis of the number of hyperlinks pointing into it (in its immediate neighbourhood) and uses the hyperlink's anchor text as an indication of the semantic content of the target document.

2. System Overview

The WT10g dataset, which was used for the small web task required some pre-processing before we were able to execute queries against it. Each individual web page had to be extracted, given a name based on its document id and saved to disk. As these files were being generated, we extracted a small amount of information from each document, which would later be used to generate intuitive results for each query, thus giving us the ability to chart our progress as we were developing the software. The information we extracted consisted of:

- Document id
- Document Title
- Document Text (< 256 bytes, to the nearest word)

In a fashion similar to our TREC-8 small web task experiments, we used an 'off-the-shelf' search engine to generate content-only results for each query. We opted to use Microsoft Index Server [3] for this purpose, though in retrospect this created more problems than it solved and consequently we are developing our own search tools for use in future experiments. This utilisation of Index Server required the extraction of each individual web page to disk and the construction of large hub files that allowed the Index Server crawler to traverse the graph of web pages from just one root page. While extracting the files to disk, we removed any additional TREC mark-up from the beginning of the document, leaving only the raw HTML of the document. This whole process took about 48 hours to complete using two computers, a PIII server with 104GB-disk space available for data & index storage, and a PIII Workstation for processing the dataset source files.

The Connectivity Data was stored in a Microsoft SQL Server 7 [4] database running on a second PIII workstation, which acted as our Connectivity Server. For a detailed description of the components of a large Connectivity Server see [5]. We maintained separate tables for inLinks and outLinks, consisting of source node id and target node id pairs for each link. The Connectivity Server worked by accepting a document id and returning a set of all inLink and outLink document ids. This approach allowed us to send about 1,000 queries per second to the Connectivity Server, which although slow, proved sufficient for our purposes.

We used the first PIII workstation again to process the queries, and calculate the Connectivity Scores for each document and generate the results. All necessary code was written in JAVA (version 1.2) for Windows NT 4. We networked the three computers together using a dedicated 100 Mbit/s switch. For an overview of the System setup see Figure 1.

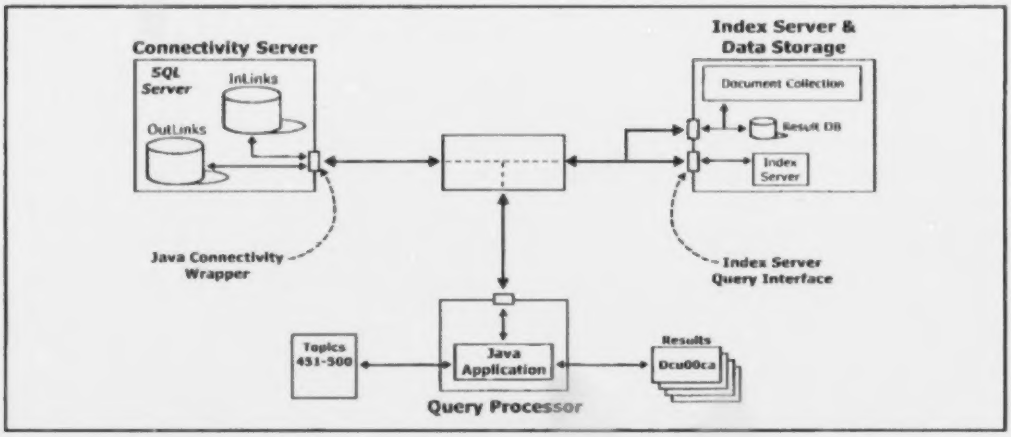


Figure 1: System overview

BEST COPY AVAILABLE

3. Experiments

Our experiments, as previously mentioned, were devised so that we could evaluate non-iterative approaches to connectivity analysis. We submitted four runs for evaluation purposes, one content-only run (*dcu00ca*), as provided by Index Server and three linkage-based runs (*dcu00la, dcu00lb, dcu00lc*). Only *dcu00lb* did not contribute documents to the assessment pools. The content run was executed before any of the linkage-based runs were executed as the basic output of Index Server was used as input into the three linkage runs.

3.1 Queries

Our queries were manually generated queries, a list of which, are included in the Appendix. In most cases the manual queries are simply the unmodified topic titles. A research student generated the queries after reading the title, description and narrative. In addition we generated a run based on automatic queries taken from the titles of each topic and have included details of where to get our results, as generated by trec_eval, in the Appendix. The automatic query run was not one of our official runs.

3.2 Content Experiment

As was previously mentioned, our experiments consisted of a two-stage process. The first stage was to generate results for a content-only run and the second was the linkage analysis stage, which reranked the set of documents returned by the content-only run. The content-only stage involved sending the query to Microsoft Index Server and extracting the results. We retrieved up to 2,000 result documents from Index Server using the Vector Space query model (there are a number of query models available). These 2,000 documents were ranked by Index Server according to their degree of relevance, but they were not scored, so we had to generate our own scores. If there were not 2,000 relevant documents returned, we processed the number of documents that were available. We assumed that these returned documents represented a large set of documents that could be considered relevant to the query, although this was not always the case as no query had more than 519 relevant documents (from results of the manual-runs). We refer to this ranked set of documents as the 'relevant-set'. In order to generate the content-based run, the top 1,000 results (where that number were available) were extracted for each query and submitted as run (*dcu00ca*) which was our *baseline* result. We generated a score for each document in the relevant set, which provided us with a content-only score for each document; it is this score that will be used in later linkage experiments.

Assuming N is the total number of documents in the result-set and R is the ranked position of that document the formula to generate the score Sc_n for each document in the 'relevant-set' is as follows:

$$Sc_n = \sqrt{\frac{N - R_n}{N}} \qquad for(R_{1...N})$$

Thus far, our *baseline* result has been gathered using a similar approach to Kleinberg's when building a root-set in HITS (see [6]). However Kleinberg only takes the top 200 documents returned from a search engine (in his case AltaVista), which he calls the root-set and expands this root-set to include all neighbouring documents, which is referred to as the expanded-set. However the root-set expansion phase of HITS seems to lead to topic-drift problems as outlined in [7], where the documents that are ranked highest are often generalisations of the topic represented by the query. In order to avoid this problem it was decided to just retain the top 2,000 documents as representing a set of relevant documents and not incorporate the neighbourhood documents to generate an expanded set. We did experiment on using the Kleinberg style expanded-set method but found that even on comparing this to the top 1,000 documents (the basis of our content-only run), we lost on average 5.68 relevant documents per query. See Figure 2 for the total recall figures summed over all queries generated using three alternative methods of generating a set of relevant documents for linkage-based processing. The approach referred to as 'base 200' is simply the top 200 documents returned from a content-only run and the 'expanded set' is generated using the HITS technique mentioned above. The 'total possible' figure is the maximum summed recall over all queries.

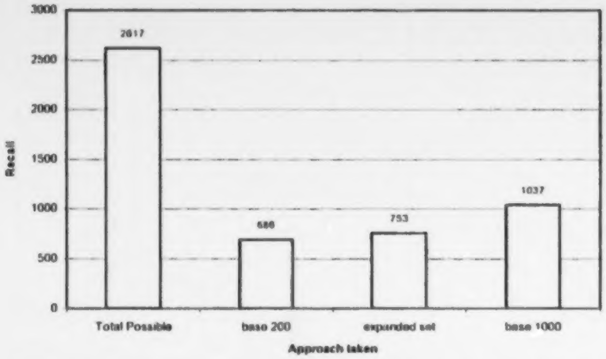


Figure 2: Recall at three different approaches to relevant-set generation

Notwithstanding any improvement in recall attained by using a large base set of documents generated using a content-only method, there are associated drawbacks. Expanding the root-set along the links does produce an expanded-set that naturally contains a high number of interconnected documents whereas selecting the top-ranked 1,000, or 2,000 documents (as we do) produces a set of documents having a much sparser set of interconnections. We may find that the use of an expanded-set is better for connectivity analysis as the expanded-set is guaranteed to have a denser set of links among the documents. This issue requires further research, but which needs to be accomplished on a new dataset.

3.3 Linkage Experiments

Our Linkage experiments were all executed at query time, and based on reranking the relevant-set of documents which were generated during the content-only stage outlined above. We developed three linkage-based approaches for our experiments, all of which adhered to the following requirements:

- Must provide a useful and accurate connectivity score for a document
- Must not require enough processing to adversely affect the performance of the search engine were it implemented in a real world system
- Must be scaleable to realistic sized datasets. We will explain below how we developed for the WT10g sized dataset but all algorithms must be capable of being implemented on more realistic datasets.

The first point is obvious, however the second point is rather more interesting. Looking at Google [8], which is visibly the most successful proponent of connectivity-based web search, it works by calculating a query independent connectivity score for each document in one processing run after all documents have been indexed by the system. This connectivity score, referred to as the PageRank [9] of a document, is then available for use by the system for all queries as part of the ranking formula, with no necessity to do any additional processing at query time.

The other widely known approach is the previously mentioned HITS, which generates linkage scores for documents at query-time. Currently the amount of processing involved in implementing HITS on a real world system would be prohibitive due to the iterative nature of the algorithm. While PageRank requires a similar iterative process, although on a vastly larger document-set (1,326,920,000 web pages as of February 2001), this is only done once per index update, but HITS requires it once per query. The algorithms outlined in this article do not have an iterative process involved and only one run through each document to be reranked is required, which helps our approaches to adhere to the second requirement above.

When discussing hyperlinks, we cannot assume all hyperlinks to be equal in value for our needs. An author writing a WWW document will create semantically different types of hyperlinks between documents, even though HTML supports only one syntactic type of hyperlink. In fact web page authors will most probably not be aware of the significance of the different link types that they are creating. In [10]

Spertus discusses hyperlinks and varieties of hyperlink information, based on information mined from identifying the target of each link. Generally speaking, on the WWW we can separate links into one of two broad types based on their intended function when being created:

- **Structural** links that link separate documents within a particular domain. They exist to aid the user in navigating within a domain, or web site and consequently cannot be seen as a source of authority judgements. See [10] for a more detailed discussion of structural links and their uses.
- **Functional** (content, or outward) links on the other hand link documents in different domains (across web site boundaries). They can be seen to mostly link from a source document to a target document that contains similar and, in the author's opinion, useful information, quite often this information is related to the concept explored in the source document.

When extracting information from hyperlinks on the WWW, we assume two properties inherent in hyperlinks from [7], these are:

- A link between two documents implies that the documents contain related content
- If the documents were authored by different people, then the first author found the second document valuable.

Because of this in the course of our research we are mainly interested in functional links as opposed to structural links and in describing our experiments we always extract only the functional links from the Connectivity Server, unless this is specifically described otherwise.

Our first connectivity analysis run (*dcu00la*) was a modification to basic citation or inLink counting. As mentioned above, we generally assume that the more popular a document is the more functional inLinks that document will have on the WWW. Letting φ be all documents in the domain of document n, the obvious choice for a basic popularity reranking formula would be as follows:

$$P_n = \sum_{m \to n} inLinks_n \qquad for(m \notin \varphi)$$

In this case the popularity of document$_n$ (P_n) is based on the number of functional inLinks into document$_n$. We implemented a similar approach last year as an unofficial run, see [1] for more details. It is notable that a system that implements only one iteration of unweighted HITS is similar to a system which ranks pages based purely on the number of functional inLinks into them.

This year we gave each document in the relevant-set a score based on its rank within the relevant-set and then added the log of the number of (inLinks + 1) multiplied by the original relevance rank of that document so that we could limit the ranking of a document of low relevance which had an unusually large indegree (number of inLinks). In this way a highly relevant document (as decided in the content analysis phase) will receive a higher rank from any inLinks than would a document that is considered less relevant in the content-only phase. This is a very simple approach and is intended as an alternative to basic inLink or citation counting.

Recall from the *dcu00ca* run that Sc_n is the content-only relevance score for document n, which was generated in the content-only stage. We generated Sc'_n as the new score for document n and ranked the documents by this score. Letting δ be the set of documents generated in the content-only phase and utilising only functional inLinks we have:

$$Sc'_n = Sc_n + Sc_n \times Log\left(\sum_{m \to n} inlink_n + 1\right) \qquad for(m \in \partial)$$

This was calculated for the top 30 documents. During development of the software for the official runs, we had experimented with reranking a variety of cut-offs for the top-ranked documents and kept the value of 30 as we found that this value produced the best results. In fact this run produced the best results of all the linkage-based runs. This would concur with the findings of AT&T in [11] which found that simple indegree ranking performed at least as well as HITS and PageRank style algorithms. This was submitted as run *dcu00la*.

Our second connectivity-based run (*dcu00lc*) attempted to improve a document's score if it was pointed at by another document, which was in itself considered to be relevant to the topic represented by the query. This is a relatively simple approach based on the previously stated assumption that a link normally exists between two documents with related content. Taking this a step further, a link between two documents with tightly related content would be a 'better' link, or a link that we could weight higher than others. To this end we increased a document score (proportionally to its current score) if it contained a link from another document that exists in the relevant-set. More precisely we increased the document score by a value which is proportional to the score of the inLink associated document and in a manner similar to the way the PageRank algorithm spreads a document's rank evenly among its outLinks, we limit the score transferred from the inLink document to be proportional to the number of outLinks it has. If effect, if a document has inLinks from a number of relevant documents then its score is increased by an amount proportional to:

- its own relevance score
- the relevance score of the inLink document
- the number of outLinks originating from the inLink document

Recall that all documents have received a score in the content only phase. The formula for calculating each document score is shown below. Let **δ** be the relevant set of documents, therefore:

$$Sc'_n = Sc_n + \left(Sc_n \times \sum_{(m \rightarrow n)} \frac{Sc_m}{\sum outlinks_m + 1} \right) \qquad for(m \in \delta)$$

This was calculated for the top 250 documents in the result-set. Once again, this figure can be changed as seen fit, but 250 is our best parameter cut-off point as found when running our experiments.

An advancement on this approach was submitted as our third run (*dcu00lb*), once again calculated for a subset of the top documents in the result-set. This viewed the inLinks to a document as hub documents. Recall from [5] that Kleinberg describes documents in terms of hub documents and authority documents with hub documents acting as a source of links into similar documents while authority documents are seen as sources of authority on a topic and are gathered together into cohesive communities by groups of hub documents. This algorithm (for *dcu00lb*) worked with authority documents in the immediate neighbourhood of the inLinking documents to approximate the identification of documents from within a well-connected community. For example, a relevant document should have a number of hub documents linking into it. Since we assume that hub documents link together groups of related documents, this hub document should be a source of links into other documents that are considered relevant to the query, and consequently they should exist in the relevant-set generated in the content-only stage. Of course we know this not always to be the case and this can be seen by viewing Figure 5 which shows our recall figures for *dcu00ca*, compared to the best, median and worst.

We augment this theory by only allowing outLink documents to be considered if they are part of the relevant-set generated during the content-only phase. This being the case, if a document is considered to be part of this set of relevant documents, it would be more useful to the user making the query. It is, in effect, an attempt to rank highly documents that are related to other relevant documents by sharing a hub document. In this approach we did not exclude hub documents, which were not included in the root set, rather we wanted to reward an inLink from a content-relevant hub document more than from a non-content relevant hub document. To that end, if a hub document is included in the relevant-set, its score (multiplied by a constant α of value 0.45), generates a preliminary score for the hub document before the outLink score is generated. We propagate the relevancy score (multiplied by a constant β of value 0.35) of each relevant authority document linked to the hub document via a functional outLink back to the hub document. The consequence of this is that the hub document now has a score which reflects its own relevance as well as that of its outLink documents. Finally this hub score is divided by the total number of outLinks from it, as was the case with *dcu00lc*. It is this score that is added to the Sc'_n score of the document being reranked. The current values for α and β were best-parameter values that we arrived while running the experiments (we had tried numerous values in the range from 0.0 - 1.0) and we plan to look again at these values on a new web dataset.

Let β be a constant to limit the score being transferred from the target document of the link to the hub document and α be a constant to limit the score being transferred from the hub document to Sc'_n during calculation of the hub document score (HSc'_n), giving:

BEST COPY AVAILABLE

$$HSc_m = Sc_m \times \alpha + \sum_{m \to p} Sc_p \times \beta \qquad for(p \in \delta)$$

or if the hub document is not in the relevant-set:

$$HSc_m = \sum_{m \to p} Sc_p \times \beta \qquad for(p \in \delta)$$

Finally, the Sc'_n is generated from the original score Sc_n and all hub scores HSc_m:

$$Sc'_n = Sc_n + \sum_{m \to n} \frac{HSc_m}{\sum_{m \to p} p + 1}$$

The final Sc'_n score is used to rank the documents for the run. We had looked into implementing both HITS and PageRank to compare our experiment's effectiveness against, but felt that the connectivity data was too sparse to be of much benefit in these cases. This was shown to be correct by the results of other participating groups that took these approaches.

4. Results

Of the four approaches we submitted, *dcu00ca*, which is the content-only run, attained highest (or equal) precision across virtually all standard rank positions. Since all other results were dependent on the quality of *dcu00ca* and didn't involve the implementation of any hyperlink-based expansion measures on the result set we were not able to produce any improvement in overall recall at rank 1,000, except in theoretical cases where a document which was not in the top 1,000 and could possibly be reranked into the top 1,000. However due to the lack of useful linkage data which became apparent during development we found that limiting the number of documents reranked would produce relatively better results, so any improvement in recall turned out to be impossible due to these limitations.

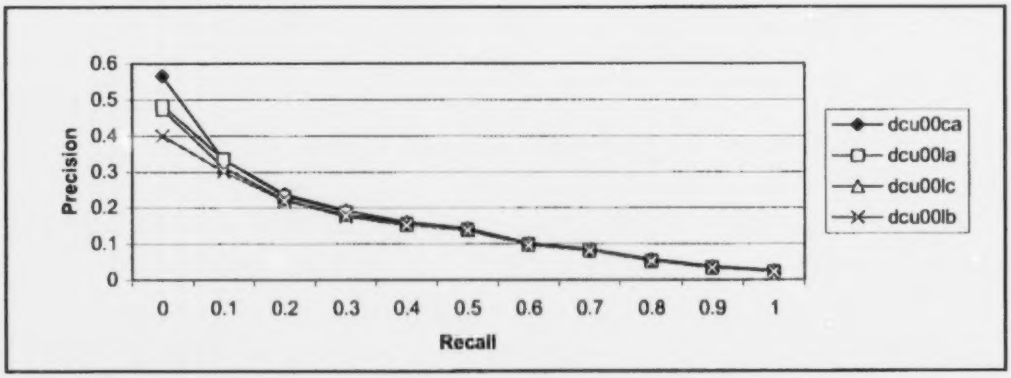


Figure 3: Precision vs. recall graph for all four runs

It would be a worthwhile exercise to see what kind of improvements could be gained by executing our experiments on a new dataset, perhaps one that has been generated with a view to conducting experiments into connectivity analysis. We may find it to be infeasible to limit our connectivity-based processing to only inLink or outLink documents that are considered directly relevant to the original query.

Another side effect of the sparse nature of the connectivity data, there was very little room for improvement over the values already in *dcu00ca* (see Figure 3 for the precision/recall graph of all four runs). With the exception of one query (486) *dcu00ca* performed equally as well, or better than all linkage based approaches. Quite often when *dcu00ca* was found to perform equally as well as the linkage approaches this was due to a lack of linkage data, which left no opportunity to rerank the documents. For details of our average precision results for each of the four runs as well as the best, median and worst overall see Figure 4 below. As you can see, the four runs have produced very similar average precision figures across all the queries. This we feel is as a result of the sparsity of connectivity data available and our best-parameter constants that limited the number of documents reranked by the linkage algorithms.

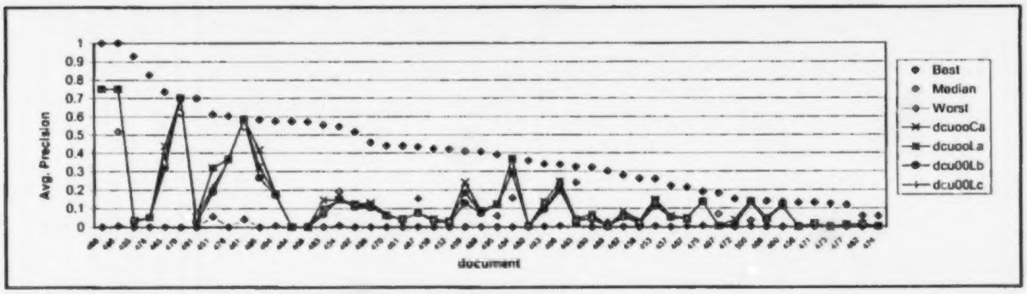


Figure 4: Average precision per topic (ordered by Best)

Our recall figures were dependent on the quality of the relevant-set generated in the content-only phase. Figure 5 shows recall at 1,000 documents for *dcu00ca* and the best, median and worst results. Recall that we never reranked even the top 1,000 documents from the relevant-set, so this recall at 1,000 figure never changed from *dcu00ca* for any of the connectivity-based runs. Hence *dcu00ca* is considered representative and is the only run plotted on the graph.

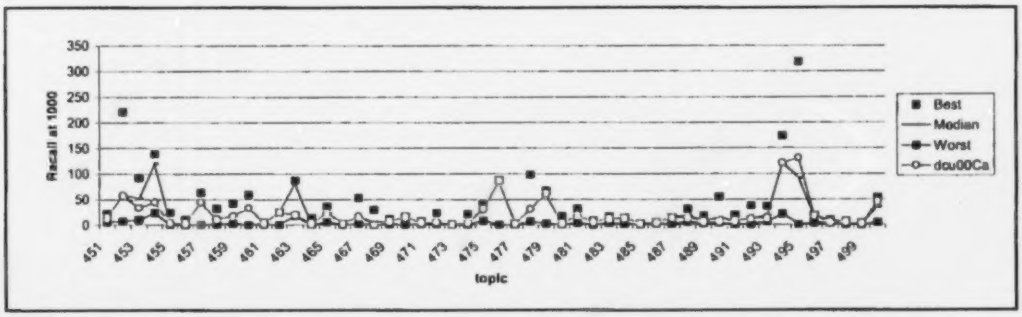


Figure 5: Content-only (*dcu00ca*) recall at 1,000 documents, including Best, Median and Worst

5. Conclusions & Future Work

It is very difficult to draw any concrete conclusions from our experiments. We feel that this is due to the fact that the WT10g dataset does not contain the density of inter-domain links as would be needed to draw these conclusions. We ran some simple experiments on the connectivity data to judge the sparseness, or otherwise, of the links within the connectivity data. We found that, in all approximately, 2% of the links were functional links while a large 98% were structural links. Recall that we were only working with functional links in our experiments. This lack of functional links seriously hampers our experiments, so much so that we decided not to implement HITS or PageRank on the dataset as additional runs. We still expect *dcu00la* would be the best reranking approach, unfortunately this is no improvement over *dcu00ca*. This does leave questions as to whether this would be best because it has the least effect on the content-result, or because the algorithm may be a more effective algorithm.

Perhaps HITS style expansion would be a better alternative than our approach to 'relevant-set' generation. The HITS approach would have the benefit of generating a set of documents that should have a higher density of functional links linking them together. Maybe a weighted HITS approach to generating a 'relevant-set' would be best. In [7] Henzinger & Bharat generate their expanded-set in the normal HITS way and then implemented a weighted algorithm over this set. This is open to additional experimentation. If we could generate a more focused 'relevant-set' of documents then we could perhaps defeat the problem of topic-drift.

In order to provide a framework within which we will be able to test out these concepts we have begun the development of our own crawler (and VSM-based IR system) to gather a dataset of language dependent documents for use in our future connectivity analysis experiments. We feel that a small minority language on the web may have an interesting link structure, in that we expect a large degree of connectivity between documents in the minority language. In developing the queuing function for selecting the candidate documents for crawling, we regard the following points as being of utmost importance:

- maintaining a weighted queue of known URLs to be indexed, favouring inter-domain links
- utilising a source of starter URLs from a source of functional outLinks (e.g. Yahoo! [12])
- documents from domains that have not yet been indexed and in the URL queue must be weighted with a highest possible weighting to increase the overall number of both functional links and domains indexed.

This would allow us to utilise a different dataset, which we hope will more accurately reflect the structure of the WWW, which is essential for us to be able to draw any concrete conclusions from our experimentation. We hope to present some of our work at TREC-2001.

6. Appendix

Due to the fact that our queries were manually generated we have made our queries available for downloading by interested parties. For the most part, the queries are unmodified from the actual topic titles. They can be found at the following URL:

http://www.compapp.dcu.ie/~cgurrin/trec9/queries.html

We are also making the results of our title-only unofficial run available for downloading. If you are interested in getting our results they can be found off the following URL:

http://www.compapp.dcu.ie/~cgurrin/trec9/titlerun.html

This page also contains links to the homepage of any third party software that we used during our experiments.

7. References

[1] C. Gurrin, A.F. Smeaton. - "A Connectivity Analysis Approach to Increasing Precision in Retrieval From Hyperlinked Documents"
Proc.8th Text Retrieval Conference (TREC-8), 1999

[2] Y. Li "Toward a more Qualitative Search Engine"
IEEE Internet Computing, Vol 2 No. 4, Jul-Aug 1998

[3] Microsoft Microsoft Index Server
http://www.microsoft.com/NTServer/web/exec/feature/ ... IndexServerSummary.asp

[4] Microsoft Microsoft SQL Server 7
http://www.microsoft.com/sql/default.htm

[5] K. Bharat, A. Broder, M. Henzinger, P.Kumar, S. Venkatasubramanian
"The Connectivity Server: fast access to linkage information on the web"
Proc. 7th International WWW Conference, 1998

[6] J. Kleinberg "Authorative Sources in a Hyperlinked Environment"
Proc. 9th ACM-SIAM Symposium on Discrete Algorithms, 1998

[7] K. Bharat & M. Henzinger - "Improved Algorithms for Topic Distillation in a Hyperlinked Environment"
Proc. 23rd Annual International ACM SIGIR Conference on Research and Development in IR. 2000

[8] Google Google Search Engine
http://www.google.com

[9] L. Page "The PageRank Citation Ranking: Bringing Order to the Web"
Stanford Digital Libraries working paper, 1997-0072

[10] E. Spertus "Parasite: Mining Structural Information on the Web"
Proc. 6th International WWW Conference, 1997

[11] B. Amento, L. Terveen, W. Hill - "Does 'Authority' Mean Quality? Predicting Expert Quality Ratings of Web Documents"
Proc. 23rd Annual International ACM SIGIR Conference on Research and Development in IR. 2000

[12] Yahoo! Yahoo! Search Engine
http://www.yahoo.com

Training Context-Sensitive Neural Networks With Few Relevant Examples for the TREC-9 Routing

Mathieu Stricker [*,**] Frantz Vichot [*] Gérard Dreyfus [**] Francis Wolinski [*]

* Informatique-CDC – Groupe Caisse des Dépôts
Direction des Techniques Avancées
4 rue Berthollet
94114 Arcueil cedex - France
{forename.surname}@caissedesdepots.fr

** ESPCI
Laboratoire d'Electronique
10 rue Vauquelin
75005 Paris, France
{forename.surname}@espci.fr

1 Introduction

The present paper describes our second participation to the routing task; it features improvements over our previous approach [Stricker *et al.*, 2000]. Our former model used a "bag of words" for text representation with a feature selection, and a neural network without hidden neuron (i.e. a logistic regression), to estimate the probability of relevance of each document. This approach was close to the ones proposed by [Schütze *et al.*, 1995] or [Wiener *et al.*, 1995] but its original feature was the use of very few relevant features for text representation (25 features per topic on the average for the TREC-8 Routing).

In this paper, two main improvements are proposed:

- The feature selection defines target words for which vectors of local contexts are subsequently defined. These vectors help disambiguate the target words and are defined by an analysis of both the relevant and the irrelevant documents of the training set.
- This new representation requires large neural networks, which are therefore prone to overfitting. A regularization technique is applied during training to favor smoother network mappings, thereby avoiding overfitting. This was achieved by adding a weight decay term to the usual cost function.

This approach led to good results on the MeSH Sample topics (S2RNsamp) and on the OHSUMED topics (S2RNr1 and S2RNr2).

2 Problem and data description

The corpus for TREC-9 is the OHSUMED collection, which is a subset of the MEDLINE database.

This corpus features documents from medical journals of years 1987 to 1991, which usually have titles and abstracts; some of them, however, only have a title. The documents were manually indexed using subject categories (Medical Subject Headings or MeSH). All documents contain the assigned MeSH headings, which are manual annotations (called .M field in the documents).

There are three topic sets for the TREC-9 routing:

1. 63 topics from the original OHSUMED query set.
2. 4904 topics based on MeSH categories.
3. 500 topics chosen amongst the 4904 previous ones called *MeSH sample topics*.

The manual annotations could be used with the OHSUMED queries (as long as it was mentioned) but NOT with the MeSH queries (nor of course with the MeSH sample queries).

We submitted 2 runs for the OHSUMED queries (one without manual annotation and one with manual annotations) and one for the MeSH sample topics.

The 1987 OHSUMED collection is intended for training and contains 54,710 documents; the test set is the 1988-91 OHSUMED collection and contains 293,882 documents. The test set is just used for evaluation and is not used in any way for building the profiles.

For each topic, a set of relevant documents is available, and all other documents are considered irrelevant. Figure 1 presents statistics for the training set for each topic set.

	OHSUMED queries	MeSH Sample queries
Number of queries	63	500
Average number of relevant documents available for training	10.6	46.5
Median	8	25

Figure1: Figures for the training set

We may observe that the number of relevant documents available per topic is small, especially for the OHSUMED queries since the median is 8.

3 Feature Selection

Each document of the collection is first tokenized into single words, case being ignored. In the following, each word is considered as a single unit called feature. No stemming is performed.

The goal of feature selection is to define, for each topic, a vector of features that are neither too frequent nor too rare, and are typical of the relevant documents. The choice of these features must be done very carefully since the quality of the filter relies heavily on this choice, irrespective of the model. These features must be chosen so as to allow a classifier to discriminate between relevant and irrelevant documents. Their number results from a tradeoff between two requirements: the larger the number of features, the larger the number of examples required to have a significant estimate of the classifier parameters; however, discarding features leads to information loss.

For each topic, a ranked list of features is computed, in which the top features are specific to the relevant documents. The method has already been used for the TREC-8 routing [Stricker *et al.*, 2000] and is discussed in detail in [Stricker, 2000].

With this technique, rare and frequent words are discarded automatically; it is useful to discard rare words because it is not possible to compute reliable statistics from them, and to discard frequent words because they carry no information.

This method is fully automatic and relies only on the computation of corpus frequency for each feature. There is no need, for example, to define a list of stop words that will depend on the language.

Contrary to last year, a Gram-Schmidt orthogonalisation with a stopping criterion was not used, because the number of relevant documents per topic was too small.

The twenty-five first features were selected, and defined the target words whose specific local context will be considered in the next section.

Figure 2 shows two lists of the top 10 features obtained at the end of this step for the topics OHSU7 "young wf with lactase deficiency". The left column is the result when the manual annotations are ignored and the right one is the result when the manual annotations are taken into account.

OHSU7 (without manual annotations)	OHSU7 (with manual annotations)
lactose	lactose
lactase	intolerance
milk	lactase
galactosidase	galactosidase
malabsorption	milk
breath	galactosidases
hydrogen	breath
digestion	hydrogen
deficient	malabsorption
yogurt	digestion

Figure 2: Examples of 10 top ranked list.

We may observe in the right column the presence of the words *intolerance* and *galactosidases,* which arise from the manual annotations.

4 Determination of local contexts

Words are naturally prone to ambiguity, and can be used in many contexts. For example, the presence of the word *intolerance*, which has been selected for the topic OHSU7 (cf. Figure 2), does not imply that a text is about "young wf with lactase deficiency".

In the past, the use of dictionaries to help disambiguate words has not proved efficient for information retrieval [Voorhees, 1993]. But disambiguation with corpus-based methods has been used successfully in [Cohen and Singer, 1996] and more recently in [Jing and Tzoukermann, 1999]. The basic idea of these methods is to disambiguate words through their local context.

Therefore, in our approach, a local context of ten words (five words on either side of the word) is considered to define which words have the highest rate of co-occurrence near a target word in the training set. Actually, two context vectors are computed for each target word: one is computed from the set of relevant documents, and the other one from the irrelevant documents.

Of course, the words with the highest rates of co-occurrence near a given word are stop words, which are useless for disambiguation. Consequently, the same procedure as used to achieve feature selection is applied to give a weight to each potential context. This method has the additional benefit of discarding automatically frequent words and rare words from the context vectors.

To compute these vectors, all relevant documents available were used, and five thousands irrelevant documents were chosen randomly. The context vectors defined by the relevant documents are called *positive context vectors* and those defined by irrelevant documents are called *negative context vectors*.

Figure 3 shows examples of positive and negative context vectors for the topic OHSU7 with manual annotations. The left column shows context vectors computed from the relevant documents for five target words, and the right one shows context vectors for the same target words computed from the irrelevant documents. The contexts are taken into account only if they appear on more than two documents.

It is worth noting that the presence of the word *intolerance* with *lactose* in its local context must be in favor of relevancy for the topic OHSU7. But, if the local context of *intolerance* contains the word *glucose* instead of *lactose* the importance of the presence of *intolerance* must decrease.

Positive context vectors		Negative context vectors	
lactose		lactose	
	intolerance		
	lactose		
	milk		
	yogurt		
	digestion		
intolerance		intolerance	
	lactose		glucose
	ul		rats
	lactobacillus		diabetes
	hydrogen		fructose
	milk		cyclosporin
lactase		lactase	
	deficient		
	milk		
	lactase		
	osteoporosis		
	organisms		
galactosidase		galactosidase	
	beta		beta
	derived		detection
	microbial		
	yogurt		
	cattle		

Figure3: Examples of local context for topic OHSU7 with manual annotations.

4.1 Choosing the size of the context vector

For each target word, the local context is chosen according to several criteria: the five first positive contexts are selected if they appear in more than two documents, and the five first negative contexts are chosen if they appear on more than ten documents. The number of documents required to take into account a context is larger for the irrelevant documents than for the relevant documents, due to the fact that irrelevant documents are much more numerous.

5 Neural Networks

5.1 Definition of the architecture

The neural network architecture must reflect the representation defined above: the influence of a target word must decrease or increase according to its local context. Therefore, instead of having a single input per target word, the local context is included as indicated in the left side of Figure 4; the right side shows the entire neural network.

Each hidden neuron is a sigmoid function and the output of the network is a logistic function in order to keep the output in the range [0,1].

Thus, this architecture contains one hidden neuron for each target words i.e. twenty-five in our case as explained in section 3.

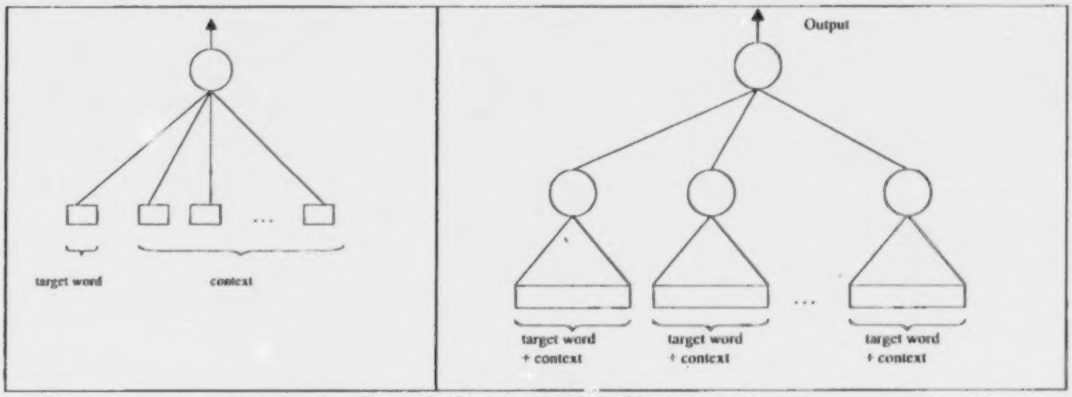


Figure 4: Neural network architecture.

5.2 Choice of irrelevant documents

Previous experiments [Stricker, 2000] have shown that it was desirable to exclude from the training set irrelevant documents for which all the target words are absent. Therefore, amongst the five thousand irrelevant documents chosen randomly, only those that share words with the relevant documents are kept.

The components of the vector are coded using the Lnu scheme defined by [Singhal, 1996].

5.3 Training with regularization

Our text representation increases the number of weights of the neural network; in addition, few relevant documents are available for training; therefore, the risk of overfitting increases, which makes the use of a regularization scheme mandatory. A penalty term is added to the usual cost function, which favors smoother functions. In our case, we use a weight decay regularization which has proven to be efficient [Krogh and Hertz, 1992][Gallinari and Cibas, 1999] and is very simple to implement.

To summarize, training is performed by minimizing a cost function G defined as:

$$G = J + \frac{\alpha}{2}\sum_{i=1}^{p} w_i^2$$

J is an of cross-entropy term appropriate for classification problems [Bishop, 1995]; the sum runs over all weights. α is a hyperparameter that defines the tradeoff between the two terms: if α is too small, the penalty term is negligible and overfitting tends to occur, whereas, if α is too large, weights will decay rapidly to zero and no training will occur.

The computation of the gradient of the new cost function is very simple since:

$$\nabla G = \nabla J + \alpha w$$

Where the quantity is computed by the well-known backpropagation algorithm.

In practice, all weights do not have the same dynamics, so that it is desirable not to use the same hyperparameter for all weights [MacKay, 1992b][Bishop, 1995]. Therefore, three hyperparameters are used according to the following relation:

$$G = J + \frac{\alpha_1}{2}\sum_{\omega \in W_0} w_i^2 + \frac{\alpha_2}{2}\sum_{\omega \in W_1} w_i^2 + \frac{\alpha_3}{2}\sum_{\omega \in W_2} w_i^2$$

W_0 denotes the bias of the first layer, W_1 denotes the weights of the first layer of weights except for the bias, and W_3 denotes the weights of the second layer plus the bias of the output.

5.4 Values of the hyperparameters

The values of (α_1, α_2, α_3) must be chosen appropriately in order to achieve a satisfactory training. A solution would be to test several values and to pick up the best ones by cross-validation. Unfortunately, this method is intractable since there are three different parameters.

A theoretical approach based on Bayesian inference has been proposed, in order to determine automatically the values of these hyperparameters during training [MacKay, 1992a]. The results of the theory rely on the estimation of integrals that cannot be computed easily. MacKay [MacKay, 1992b] has proposed several approximations in a theoretical framework known as the *evidence framework* to make the computation feasible. Unfortunately, these results did not provide good results on previous experiments.

Consequently, the values of the hyperparameters were chosen according to experience that we gathered previously on other corpuses (TREC-8 and Reuters21578):

$$\alpha_1 = 0.001 \qquad \alpha_2 = 0.1 \qquad \alpha_3 = 5.0$$

6 Results

We proposed three runs for the TREC-9 routing:

1. S2RNr1: 63 OHSUMED queries without using the manual annotations.
2. S2RNr2: 63 OHSUMED queries using the manual annotations.
3. S2RNsamp: 500 MeSH sample queries (without manual annotations).

The score is computed thanks to the *uninterpolated average precision* as described in [Hull and Robertson, 2000].

Figure 5 shows the comparisons of our runs with the other officials runs submitted for the TREC-9 Routing.

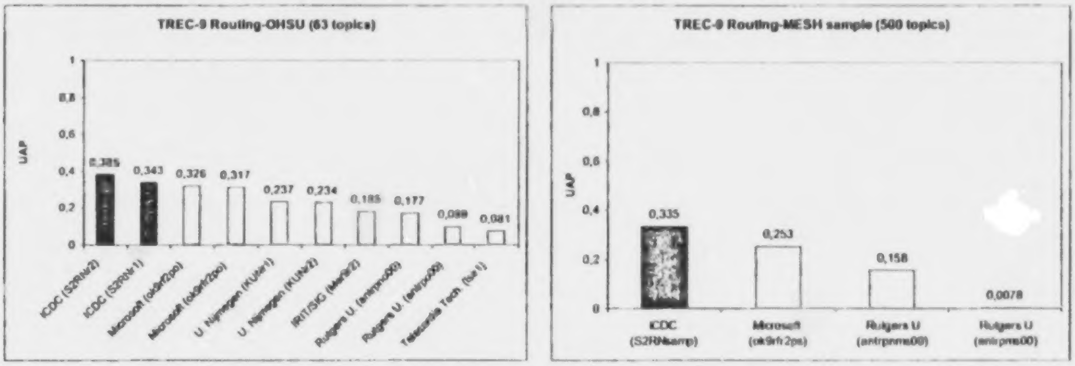


Figure 5: Comparisons of officials runs for the TREC-9 Routing.

For each subtopics (OHSU and MeSH sample), our method achieved the top scores. It is worth noting that the run S2RNr2 has better results than S2RNr1. It shows that our model can take advantage of the manual annotations without changing anything to our approach, since the difference relies only in the reading of the ".M field" which is considered as part of the text for S2RNr2.

In the case of the MeSH sample, the gap between our run and the second one is bigger than in the case of the OSHU topics ; it seems that our method has taken advantage of the greatest number of relevant documents available for training on the MeSH sample.

REFERENCES

[Bishop, 1995] C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

[Cohen and Singer, 1996] W. W. Cohen, Y. Singer. Context-sensitive methods for text categorization. *Proceedings of the 19th Annual International Conference on Research and Development in Information Retrieval (SIGIR '96)*, 307-315, 1996.

[Gallinari and Cibas, 1999] P. Gallinari, T. Cibas. Practical complexity control in multilayer perceptrons. *Signal Processing*, 74, 29-46, 1999.

[Hull and Robertson, 2000] D. A. Hull, S. Robertson. The TREC-8 Filtering Track Final Report. *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*. NIST Special Publication 500-246, 35-56, 2000.

[Jing and Tzoukermann, 1999] H. Jing, E. Tzoukermann. Information Retrieval Based on Context Distance and Morphology. *Proceedings of the 22nd Annual International Conference on Research and Development in Information Retrieval (SIGIR '99)*, 90-96, 1999.

[Krogh and Hertz, 1992] A. Krogh, J.A. Hertz. A Simple Weight Decay Can Improve Generalization. *Advances in Neural Information Processing Systems*, 4, J.E. Moody, S.J. Hanson and R.P. Lippmann, eds., Morgan Kauffmann Publishers, San Mateo CA, 950-957, 1992.

[MacKay, 1992a] D. J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3), 415-447, 1992.

[MacKay, 1992b] D. J. C. MacKay. A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, 4(3), 448-472, 1992.

[Schütze *et al.*, 1995] H. Schütze, D. A. Hull, J. O. Pedersen. A Comparison of Classifiers and Document Representations for the Routing Problem. *Proceedings of the 18th Annual International Conference on Research and Development in Information Retrieval (SIGIR '95)*, 229-238, 1995.

[Singhal, 1996] A. Singhal. Pivoted Length Normalization. *Proceedings of the 19th Annual International Conference on Research and Development in Information Retrieval (SIGIR '96)*, 21-29, 1996.

[Stricker, 2000] M. Stricker. *Réseaux de neurones pour le traitement automatique du langage : conception et réalisation de filtres d'informations*. Thèse de l'université Paris VI, 2000.

[Stricker *et al.*, 2000] M. Stricker, F. Vichot, G. Dreyfus, F. Wolinski,. Two Step Feature Selection for the TREC-8 Routing. *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*. NIST Special Publication 500-246, 425-430, 2000.

[Voorhees, 1993] E. M. Voorhees. Using WordNet to disambiguate words senses for text retrieval. *Proceedings of the 16th Annual International Conference on Research and Development in Information Retrieval (SIGIR '93)*, 171-180, 1993.

[Wiener *et al.*, 1995] E. D. Wiener, J. O. Perdersen, A. S. Weigend. A Neural Network Approach for Topic Spotting. *Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval (SDAIR '95)*, 317-332, 1995.

FDU at TREC-9: CLIR, Filtering and QA tasks

Lide Wu, Xuan-jing Huang, Yikun Guo, Bingwei Liu, Yuejie Zhang
Fudan University, Shanghai, China

This year Fudan University takes part in the TREC-9 conference for the first time. We have participated in three tracks of CLIR, Filtering and QA.

We have submitted four runs for CLIR track. Bilingual knowledge source and statistical-based search engine are integrated in our CLIR system. We varied our strategy somewhat between runs: long query (both title and description field of the queries involved) with pseudo relevance feedback (FDUT9XL1), long query with no feedback (FDUT9XL2), median query (just description field of queries involved) with feedback (FDUT9XL3) and, the last, mono long query with feedback (FDUT9XL4).

For filtering, we participate in the sub-task of adaptive filtering and batch filtering. Vector representation and computation are heavily applied in filtering procedure. 11 runs of various combination of topic and evaluation measure have been submitted: 4 OHSU runs, 1 MeSH run and 2 MSH-SAMPLE runs for adaptive filtering, and 2 OHSU runs, 1 MeSH run and 1 MSH-SAMPLE run for batch filtering.

Our QA system consists of three components: Question Analyzer, Candidate Window Searcher and Answer Extractor. We submitted two runs in the 50-byte category and two runs in the 250-byte category. The runs of "FDUT9QL1" and "FDUT9QS1" are extracted from the top 100 candidate windows. The other two runs of "FDUT9QL2" and "FDUT9QS1" are extracted from the top 24 candidate windows.

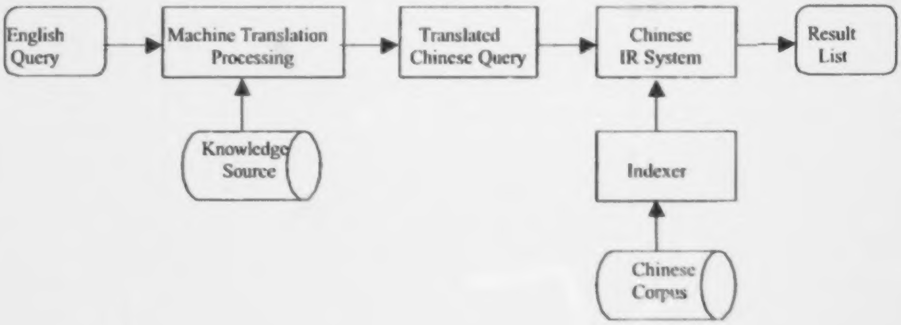
1. Cross-Language IR

We focused our attention on Chinese document indexing and query translation. All query processing was fully automatic and both long and short query translation are covered.

Our overall strategy to CLIR task is to translate English query into Chinese word list, since we feel it is not feasible to build a document translation system in such a short period, while it is much more reasonable to disambiguate word sense in context of long query by statistical approach, such as POS and knowledge. Once queries have been translated, we use IR techniques, which is a variant of MIT's approach[1] and probabilistic methods to obtain relevant document list. The whole corpus has been indexed with Chinese NLP techniques developed by our group in recent years [2]. Finally, we also explore the weight of words in both of the title and description fields.

The system infrastructure is illustrated in figure 1.1. Description of each part is followed.

Figure 1.1 System architecture for CLIR



1.1 Indexer

We explored two different indexing methods for our CLIR task, one is word-based Chinese indexing module, and the other is n-gram based indexing module. Because Chinese is different from English in that there are no extra spaces between Chinese words, we must first segment Chinese character sequence into words or n-grams in order to index documents.

1.1.1 Chinese word segmentation module

Figure 1.2 illustrates the architecture of our word segmentation sub-system. Given a document, it is first divided into a sequence of sentences (sub-sentences) by punctuation such as full stop, comma, etc. Each sentence then passes through the sentence segmentor and is segmented into a sequence of words. Finally, a text-level post-processor will act on the word sequence and generate the final segmentation result.

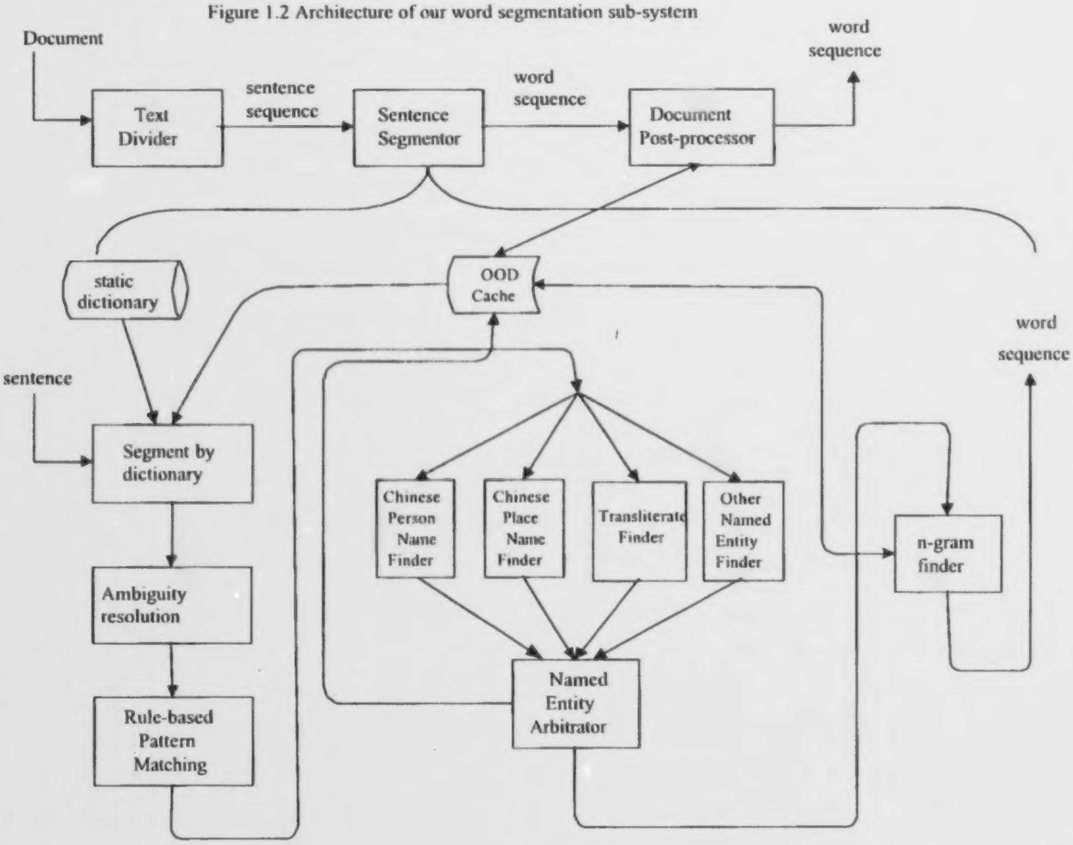
- Dictionary

Two kinds of dictionaries are used during the segmentation process. One is the static dictionary, which records Chinese lexical words and is unchangeable. The other is the OOD (out-of-dictionary words) cache, which records the newly found OOD and changes dynamically.

- Sentence segmentor

The input sentence is first segmented by both static and dynamic dictionary. Ambiguous strings are handled at the same time. We use a pattern-matching module to recognize those OODs with fixed structure pattern, such as money, date, time, percentage and digit.

Figure 1.2 Architecture of our word segmentation sub-system



Document
Text Divider
sentence sequence
Sentence Segmentor
word sequence
Document Post-processor
word sequence
static dictionary
OOD Cache
word sequence
sentence
Segment by dictionary
Ambiguity resolution
Rule-based Pattern Matching
Chinese Person Name Finder
Chinese Place Name Finder
Transliterate Finder
Other Named Entity Finder
n-gram finder
Named Entity Arbitrator

The recognition module of person's name, place, organization and transliteration is more complex. Contextual and structural information both play important roles in identifying these kinds of OODs. The former can provide external evidence for deciding word boundary and predicting the category of OOD. The latter can provide internal evidence for suggesting and validating the appearance of certain OOD. For example, in Chinese, family surnames are stereotypical. We've made lots of statistical analysis on various categories of OOD and built correspondent identification modules for each. Each module works independently. A named entity arbitrator will take effect when two or more kinds of name entities conflict with each other and select the most probable one.

Beside these OOD types mentioned above, there are still many other kinds of OODs. We can also recognize some OODs according to its string frequency and internal characters' mutual information.

- Document post-processor

During the sentence-by-sentence segmentation, some OODs will not be recognized until they occur several times. Therefore, the OOD cache changes continuously until the whole document passes through the sentence segmentor. After that, a document post-processor based on the final content of the cache is necessary to detect missed or mistakenly segmented words before.

1.1.2 n-Gram based Tokenization

We also implement n-Gram based tokenization process, which does not need sophisticated segmentation method. The document is simply cut into sequence of bigrams. We want to know whether the effectiveness of IR based on n-gram is comparable, inferior or superior to that based on word segmentation.

1.1.3 Word Indexing

Every document in the corpus is cut into no more than 64K segment to make indexing procedure more robust and normalize the document length. After being segmented, text id, term frequency, document frequency and term position are stored for the task. No stop word is removed from the invert file, since the corpus is rather small.

In order to optimize the disk space and I/O in retrieval time, we have also implemented invert file compression. The file was then decrease to about one half of its original size.

1.2 Query Translation

The essence of cross-language information retrieval is to use queries in one language to retrieval documents from a pool of documents written in other languages. This may be achieved by using query translation, document translation, or by using both query and document translation.

Here, we adopt query translation as the dominant strategy, use English query to be translated object, and utilize English-Chinese bilingual dictionary as the important knowledge resource to acquire correct translations. So by using our Chinese Information Retrieval system, the complete English-Chinese CLIR process can be implemented successfully.

1.2.1 Knowledge Source Construction

The knowledge source used in English-Chinese-oriented CLIR system mainly includes dictionary knowledge and Chinese Synonym Dictionary. In addition, stopword list and word morphological resumption list are also utilized in our system. In fact, dictionary is a carrier of knowledge expression and storage, which involves almost all information about vocabulary, namely static information.

(1) English-Chinese Bilingual Dictionary

This dictionary is mainly used in translation processing in word level and phrase level. And it consists of three kinds of dictionary component as follows:

- Basic Dictionary--A basic knowledge source independent of particular field, which records basic linguistic vocabulary;
- Technical Terminology Dictionary –Recording terminology knowledge in a particular technical field, which is mainly referred to Hong Kong commercial terminology knowledge and incorporated in the basic dictionary;

- Idiom Dictionary--Recording familiar fixed matching phenomena, such as idiom and phrase.
- The whole bilingual dictionary involves almost 50,000 lexical entries. And each entry is established as the following data structure:

English Lexical Information	Part-of-Speech Information	Subcategory Information	Concept Number	Matching Information	Semantic Class Code	Chinese Lexical Information

Two examples of particular entry representation form in dictionary are listed as the following:

*happiness || n || ng || 0 || M ;[U]; || bbaaa || 幸福 (felicity) ||||

*handle || v || vt || 5 || Wv3;T1]; || CBBC || 处理 (processing) ||||

(2) Chinese Synonym Dictionary (同义词词林)

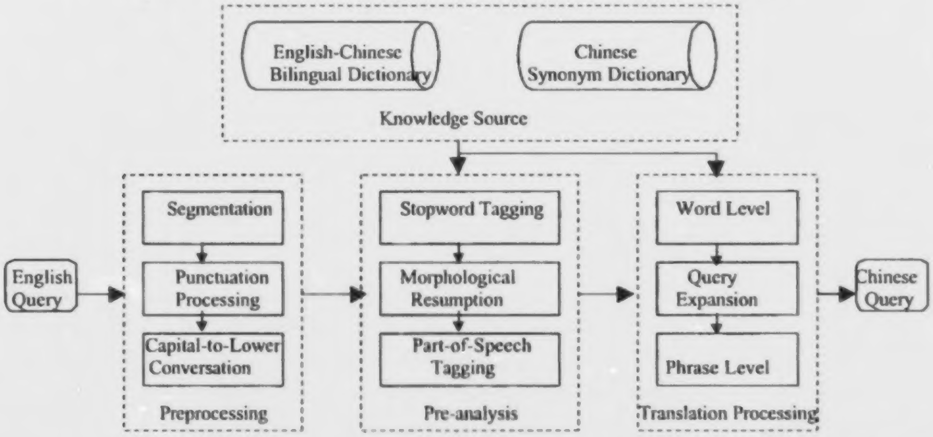
Actually, this dictionary is a thesaurus, which involves nearly 70,000 entries. All entries are arranged according to specified semantic relations. It is mainly used in expanding translation that has passed through translation processing, namely query expansion.

While the stopword list is used in tagging the stopwords in English query, and the English morphological resumption list which describes all irregular varieties about vocabulary is used in morphological resumption of words with irregular variety forms.

1.2.2 Translation algorithm

The basic framework of English-Chinese-oriented translation algorithm is mainly divided into three parts, as shown in Figure 1.3.

Figure 1.3 Basic framework of English-Chinese-oriented query translation algorithm


English-Chinese Bilingual Dictionary
Chinese Synonym Dictionary
Knowledge Source
English Query
Segmentation
Punctuation Processing
Capital-to-Lower Conversation
Preprocessing
Stopword Tagging
Morphological Resumption
Part-of-Speech Tagging
Pre-analysis
Word Level
Query Expansion
Phrase Level
Translation Processing
Chinese Query

- Preprocessing -- including sentence segmentation, punctuation tagging and capital-to-lower letter conversation for English query;
- Pre-analysis -- including stop words tagging, word morphological resumption and POS tagging processes;

Considering that translation processing is related with some stopwords, the stopwords must be tagged by stopword list. Because there are some words with variety forms in English query, translation knowledge cannot be induced correctly. So by using English-Chinese bilingual dictionary, morphological resumption lists for irregular variety and heuristics for regular variety, we get words' original form from the process called "morphological resumption". To analyze word part-of-speech, we develop a HMM-based (Hidden Markov Mode) Part-of-Speech Tagger.

- Translation processing -- including translation processes in two levels: word level and phrase level.

Word level translation: By using the basic vocabulary part of English-Chinese bilingual dictionary, this process mostly implements translation word by word. For word disambiguation, a word may correspond with several kinds of different sense. Word sense is related with particular word, and cannot be given without particular linguistics environment. The condition of linguistics environment may be syntactic and semantic parameters. When selecting a particular word, the difference mark of word should be chosen. This difference mark represents a certain syntactic and semantic feature, and identifies the sense of word uniquely, namely Concept Code. The concept code together with lexical entry can decide a certain word sense to accomplish word sense disambiguation. For machine translation, word disambiguation should be a very important problem. But in our CLIR system, in some degree, word disambiguation has not taken some obvious affect to retrieval efficiency. At the same time, in order to provide more query information to retrieval system, by using "Chinese Synonym Dictionary", expansion operation is done for translation knowledge through translation processing. According various synonymous relations described in the dictionary above, all synonyms corresponding with translation knowledge is listed, namely completing query expansion process. Thus, more affluent query information can be provided to retrieval system. So the retrieval efficiency is increased greatly, and the retrieval performance is improved.

Phrase level translation: This process is implemented based on the idiom dictionary part of English-Chinese bilingual dictionary. The recognition of near distance phrase and far distance phrase is an important problem. Here, by adopting Greedy Algorithm, the recognition and translation processing of near distance phrase is mainly completed, shown as the following:

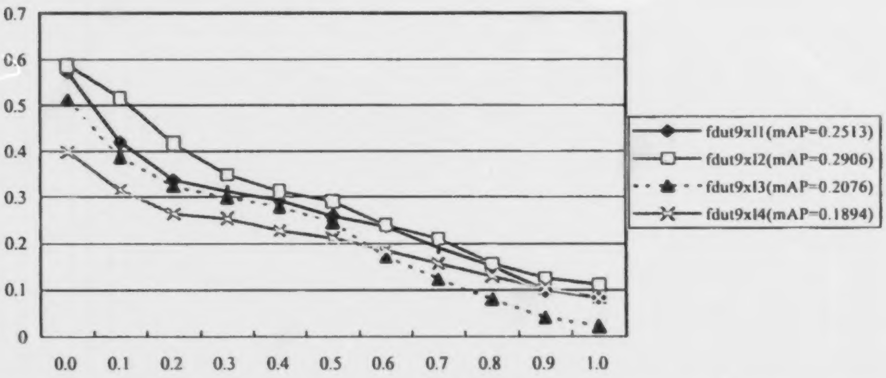
- Acquiring phrase set which have some query word as the head word of the idiom from English-Chinese bilingual dictionary;
- Identifying the phrases which have the same word as head word and the same number of word as the phrase in the above set;
- Comparing each one of the identified phrases and every member in the correspondent phrase set and finding out the matched phrase with the maximum length.

1.3 Experiment

Our search engine scores document by maximum likelihood ratio, put forward by Spoken Language Systems Group in MIT [1]. In our retrieval experiment, we use the TREC-5 Chinese task as the "training" data set for tuning and optimizing our retrieval model. Finally, our best run has achieved the mAP (mean average precision) of 0.3869, which is about the same as the best result at that time.

After that, we submit four runs for CLIR official evaluation this year. Figure 1.4 is the official precision and recall curve and the mAP score of our 4 CLIR runs. The first three of them are automatic query translation run, using our word segmentation approach for indexing, while the monolingual run we submit uses n-gram based segmentation. Although the results are not as good that of training results, the run of "fdut9xl2" still can achieve the mAP of near 0.30.

Figure 1.4 official Prec and Recall Curve and mAP score on TREC-9 CLIR Task



We have outstanding performance for automatic query translation run: most of the queries outperform the average in the run "fdut9xl1". However, the monolingual run is not as good as we expected. We speculate that it may be due to our sophisticated segmentation method, which could correctly segment the names of people, place and organization etc. In other word, indexer based on word segmentation performs much better than indexer based on n-gram.

2. Filtering

For filtering, we participate in the sub-task of adaptive filtering and batch filtering. Our research focuses on how to create the initial filtering profile and threshold and then modify them adaptively.

Our batch runs share the same adaptation module with adaptive runs. Therefore, our batch runs are actually batch adaptive runs.

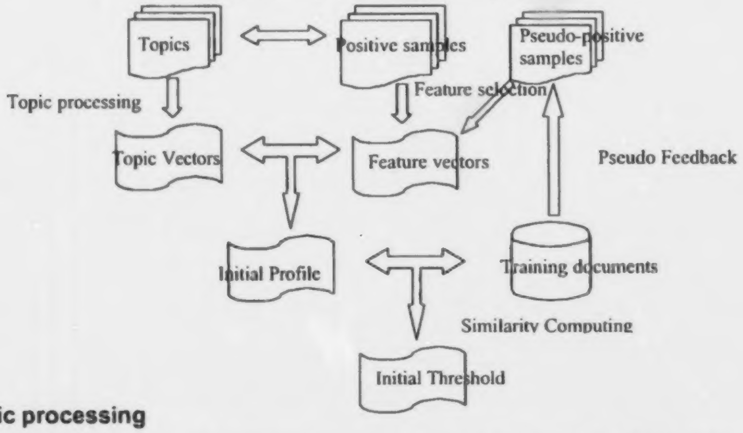
There is only a slight difference in the initialization (in other word, training) module of our batch and adaptive runs. Full relevance judgement is provided in batch filtering, while only a small proportion of relevance judgement is provided in adaptive filtering. As a result, for batch run, we can obtain a set of "Negative" documents, which are those irrelevant documents with high similarity to the filtering profile, and then make use of such documents. For adaptive run, we try to discover more pseudo-relevant documents based on the topic and limited relevance judgement in order to optimize the initial profile.

Following is the detailed introduction to our training and adaptation module of our adaptive and batch task.

2.1 Training of adaptive filtering

Figure 2.1 shows the architecture of the training in adaptive filtering. At first, topics are changed into topic vectors, while feature vectors are extracted from positive and pseudo-positive document samples. The initial profile is the weighted sum of topic and feature vectors. Then we compute the similarity between the initial profile and all training documents to find the optimal initial threshold for every topic.

Figure 2.1 Architecture of the training in adaptive filtering



2.1.1 Topic processing

Topic is being processed as such: Firstly, every word in the topic is labeled with one of four attributes: title words; description words; negative words (words which behinds the word "without"); domain dependent stopwords such as "*document*" and "*describe*". Each kind of attribute is assigned with a coefficient. If one word occurs several times in the topic and different attributes are labeled, the maximum coefficient is chosen for it. Different coefficients are chosen for OHSU and MeSH topics.

As we know, for OHSU topic, title is the description of patient, and description is the information request. Both are important. However, for MeSH topic, title is MeSH concept name and description is the definition of the concept. We have found the description part is not as important as the title. For example, the description of the concept of "*abdomen*" is that "*the portion of the body that lies between* *the thorax and the pelvis*". If we expand the initial query with such word as "*thorax*" or "*pelvis*". the performance will even be hurt.

In our experiment, the coefficients of OHSU topics are set to 1, 1, -1 and 0 respectively, while those of MeSH topics are 1, 0, 0 and 0 respectively.

The weight of each topic word is set to be the product of its coefficient multiplied with Smart's *ltc* weight: *ltc*=log(*N/n*)[3], where *N* is the total number of documents and *n* is the number of documents in which the word occurs. We adopt *ltc* weight because the simplicity in computing. We have also tried other weighting formulas with relevant information and do find they can lead to better performance when only topic information is utilized in profile creation. However, once topic vector is combined with feature vectors from the training documents to form the initial profile, more complicated weight algorithm no longer ensures better performance.

2.1.2 Feature selection

Since the total number of all words is very large and then it cost more time in similarity computation, we decide to select some important words from them. First, we use Porter's stemmer to get the root form of every word. Then we remove the stopwords and low frequent words (occur no more than 6 times in the training document sets). Then we compute the *logarithm Mutual Information* between remaining words and topics:

$$\log MI(w_i, T_j) = \log\left(P(w_i \mid T_j) \big/ P(w_i) \right) \qquad (2.1)$$

Where, w_i is the ith word and T_j is the jth topic. Higher logarithm Mutual Information means w_i and T_j are more relevant. P(w_i) is estimated by *maximal likelihood method*. Since the total number of relevant documents is very small, $P(w_i \mid T_j)$ is estimated by *Turing-Good method.*

For each topic, we select those words with logarithm Mutual Information higher than 3.0 and occurs more than once in the relevant documents. Thus the average feature number of each topic is around 100. Logarithm Mutual Information is not only used as the selection criterion, but also as the weight of feature words.

2.1.3 Creating initial profile

Each topic profile is represented by a vector which is the weighted sum of topic vector, feature vector from positive (relevant) documents and feature vector from pseudo relevant documents with the coefficient of A, B and C.

We add a procedure of **pseudo feedback** to acquire more relevant documents for training. Those documents that have highest similarity and don't occur in the positive documents are regard to be relevant.

The initial profile is created by two-phase method. First we get the pseudo relevant documents; then we get the initial profile and re-compute the optimal initial threshold.

During the first phase, A, B and C are set to be 0.4:1.0:0. We set C to 0 because we have no similarity score at this time. Then some documents are selected as pseudo positive documents. As for how many documents should be appended, we adopt two methods. The first method choose the N highest similar documents which don't occur in the positive documents; while the second method choose those documents whose similarity is higher than a fixed scale (α) of the highest similar positive document. These two method lead to similar results. Here we set α = 0.45, or N=10.

During the second phase, A, B and C are set to be 0.25:1.0:0.25. The parameter of A becomes smaller because now we have so much positive document that topic vector becomes relatively less important. Therefore, we have got the initial profile.

2.1.4 Similarity Computation

The similarity between the profile and training documents is computed by the cosine formula:

$$Sim(d_i, p_j) = Cos\theta = \sum_k d_{ik} * p_{jk} \Big/ \sqrt{(\sum_k d_{ik}^2)(\sum_k p_{jk}^2)} . \qquad (2.2)$$

Where, p_j is the profile vector of the jth topic and d_i is the vector representation of the ith document. d_{ik}, the weight of the kth word in d_n is computed as such: $d_{ik} = 1 + \log tf_{ik}$, where tf_{ik} is the term frequency of the kth word in the ith document.

Documents are first removed of the redundant tag information and then stemmed. Only the document identifier, tile and abstract are reserved, while MeSH headings and other fields are all removed.

2.1.5 Setting initial threshold

Once the threshold is set, those documents with similarity greater than the threshold are regarded to be relevant and those documents with similarity smaller than the threshold are regarded to be irrelevant. Then we can compute the evaluation criteria such as T9U and T9P under different threshold. Thus the initial threshold are set to be the threshold which can result in the largest T9U or T9P.

2.2 Training of batch filtering

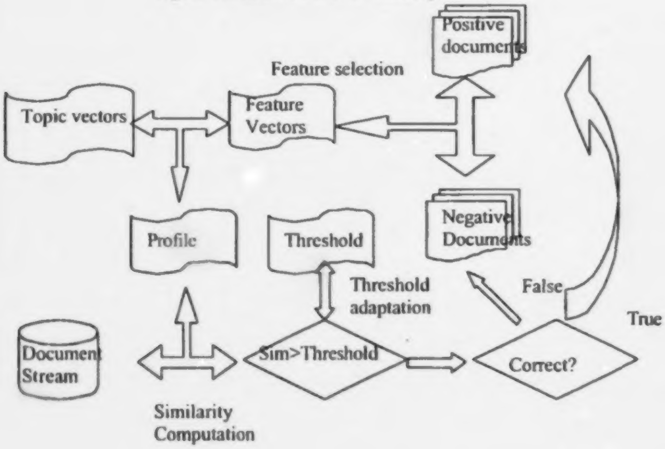
Training of batch filtering is quite similar to adaptive filtering. The only difference is that feature vectors are now extracted from positive and negative (irrelevant but with high similarity) document samples. Each topic profile is represented by a vector which is the weighted sum of topic vector, feature vector from positive documents and feature vector from negative documents with the coefficient of A, B and C.

Initial threshold is also set in a two-phase method. At the first phase, A, B and C are set to be 0.25:1.0:0. For each vector profile, we calculate its similarity with every training document and then set the temporary similarity threshold. After that, negative documents are selected to be those irrelevant documents with similarity higher than the temporary threshold and could lead to wrong judgement. During the second phase, A, B and C are set to be 0.25:1.0:-0.25.

2.3 Adaptation

For adaptive and batch filtering we adopt the same adaptation procedure. Figure 2.2 shows the architecture for the adaptation. For each document in the stream, its similarity with the specific topic profile is computed. If the similarity is greater than the threshold, it is assumed to be relevant. Then we search the "qrel" file to see whether it is really relevant and do some adaptation accordingly.

Figure 2.2 Architecture for the adaptation



2.3.1 Adaptation of threshold-T9P

Thresholds are adjusted after β documents have been processed (for this experiment, β=8000). For different evaluation measure of T9P and T9U, the adaptation is also different.

In order for the optimization of T9P, the purpose of threshold adaptation is to make sure that about 50 documents are retrieved during 4 years. Therefore M documents should be retrieved in the β-document interval. For each topic, we define:

Cor: # of documents correctly retrieved in the interval
Rtv: # of documents retrieved in the interval
Cor1: # of documents correctly retrieved heretofore
Rtv1: # of documents retrieved heretofore
M1: # of documents should be retrieved heretofore
T: Similarity threshold
Algorithm:
If Cor<Rtv*0.20 && Rtv>max(M,4) , then T*=1.2
    (If the precision is too slow, the threshold should be increased quickly)
If Rtv>M && Rtv1>M1, then T*=1.1
    (If documents are retrieved more than required, the threshold should be increased)
If Rtv<M && Rtv1<M1, then T*=0.9
    (If documents are retrieved less than required, the threshold should be lowered)

We have supposed that we can retrieve fewer documents at first and then retrieved more documents after profiles are updated. However, such experiment cannot lead to better results.

2.3.2 Adaptation of threshold-T9U

In order for the optimization of T9U, the purpose of threshold adaptation is to make sure that documents should be retrieved with high accuracy. But if the precision is too high, thresholds should also be decreased to retrieval more documents and then get larger T9U.

Algorithm:
If Cor<Rtv*0.10 && Rtv>max(M,4) , then T*=1.2
    (If the precision is too slow, the threshold should be increased quickly)
If Rtv-1>M && Cor+1>Rtv*0.33, then T*=1.1
    (If enough documents have been retrieved and precision is too low, the threshold should be increased)
If Rtv<M && Cor-1>Rtv*0.25 or Cor-1>Rtv*0.33 or Cor=0, then T*=0.9
    (If documents are retrieved less than required with moderate precision, or precision is too high, or no document is retrieved, the threshold should be lowered)

In addition, if two irrelevant documents are retrieved continuously, T*=1.1.

Here, M represents the number of documents should be retrieved in the β-document interval. However, adaptive filtering systems cannot take into account the percentage that are relevant over the entire test set for a particular query in building their retrieval rules. Under such condition, M is estimated from the training corpus while relevant and pseudo relevant documents are taken into account. Although M is actually variable among different topics, we just use the average value for the convenience for computation.

2.3.3 Adaptation of topic profile

Once a retrieved document has been judged to be relevant, it is added to the positive document set for further adaptation, otherwise it is added to the negative document set. During profile adaptation, feature vectors are extract from positive documents and negative documents. The new topic profile is the weighted sum of topic vector, feature vector from positive documents and feature vector from negative documents. Thus not only the weight of features but also the feature words can be adjusted. The coefficient of A, B and C are still 0.25, 1.0, and -0.25.

Since relevant document is too scarce, we adjust the topic profile only after β*4 documents have been processed. In fact, after processing β document, adaptation is triggered. Among 4 successive adaptation, the first 3 are threshold adaptation and the last one is profile adaptation. We don't adjust threshold and profile simultaneously because the threshold is optimized for the original profile.

2.4 Evaluation results

This year Fudan University has submitted 11 runs for adaptive filtering and batch filtering. We submit no routing runs. Table 2.1~2.3 summarize our adaptive and batch filtering runs.

Table 2.1 shows the results of OHSU topics. The "*score*" column is the score of each run under different evaluation measure. Micro recall and precision are calculated globally for all the topics, while macro recall and precision are averaged across all the topics[4]. The last columns give the number of topics in which our runs perform better, equal and worse than median ones. And the numbers inside the parentheses shows the number of topics in which our runs perform best.

Task	*Measure*	*Run*	*Score*	*Recall*		*Precision*		*Comparison with median*		
				Micro	*Macro*	*Micro*	*Macro*	*>(Best)*	*=*	*<*
Adaptive	T9U	FDUT9AF2	9.6	0.212	0.181	0.473	0.319	51(7)	6	6
	T9P	FDUT9AF1	0.264	0.277	0.300	0.283	0.271	37(6)	9	17
		FDUT9AF3	0.265	0.276	0.301	0.286	0.273	39(5)	8	16
		FDUT9AF4	0.249	0.263	0.285	0.278	0.259	34(9)	2	27
Batch	T9U	FDUT9BF1	13.6	0.276	0.245	0.492	0.390	37(20)	11	15
	T9P	FDUT9BF2	0.317	0.331	0.379	0.326	0.322	45(21)	7	11

Table 2.1 Adaptive and batch filtering for OHSU topics

Table 2.2 and 2.3 show the results of MeSH and MeSH sample topics. Our MeSH and sample runs don' t perform as good as OHSU runs.

Task	*Measure*	*Run*	*Score*	*Comparison with median*		
				>(Best)	*=*	*<*
Adaptive	T9P	FDUT9AF6	0.356	134(134)	148	218
	T9U	FDUT9AF7	29.3	120(85)	72	308
Batch	T9P	FDUT9BF3	0.430	169(61)	151	180
	T9P	FDUT9BF4	0.440	215(101)	138	147

Table 2.2 Adaptive and batch filtering for MeSH sample topics

Task	*Run*	*T9P*	*Comparison with median*		
			>(Best)	*=*	*<*
Adaptive	FDUT9AF5	0.351	1297(1297)	1072	2535
Batch	FDUT9BF3	0.418	2297(2297)	0	2607

Table 2.3 Adaptive and batch filtering for MeSH topics

3. Question Answering

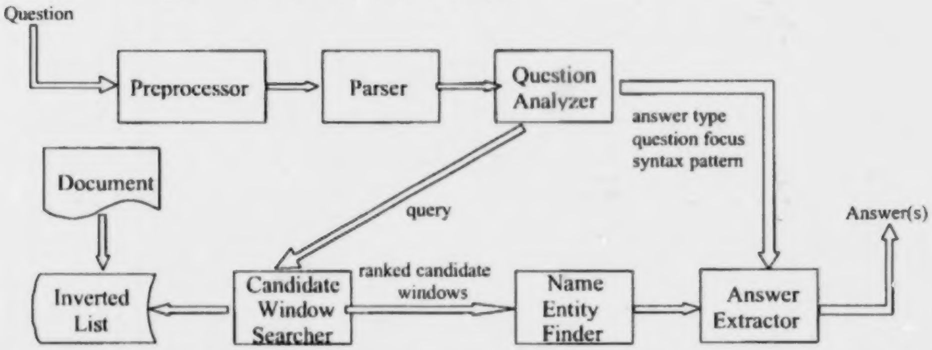
Question Answering is an interesting challenge for NLP researchers because it requires a combination of many traditional NLP techniques, such as tokenization, parsing, named entity identification and retrieval.

The next section introduces Fudan TREC-9 question answering system. It is followed by the detailed discussion of three main components. Followed are the evaluation results. Finally we will discuss the future prospects of our system.

3.1 Overview of Fudan Question Answering System

Similar to other systems[5], our system consists of three components: *Question analyzer, Candidate Window Searcher* and *Answer extractor*. The architecture is illustrated by Figure 3.1.

Figure 3.1: Architecture of the Fudan QA Systems



Initially, with a question parser and semantic mapping chart, we process the given questions and extract useful information: answer type, question focus and the syntax pattern of question. Furthermore, the Question Analyzer generates a set of query terms.

Each document retrieved by our ranked Boolean search engine is divided into segments of about 4k-byte. Each candidate segment is assigned with a score according to its similarity to the query generated by question analyzer.

The top-ranked segments are then passed to the Answer Extractor. A named entity finder based on HMM model[6] and a syntax parser based on chart algorithm are involved in this procedure. The extraction and ranking of the final answer are based on some empirical feature matching.

3.2 The Question Analyzer

The *Question Analyzer* attempts to excavate all available information inside the given question and generate a query for search engine.

In order to extract the real answer from the tremendous collection of documents, it' s very important to know what the question is asking for. Fortunately, quite a few questions request certain type of answer. For example, for the question "*Who invented the paper clip?*", a person name is needed. We can either judge the question' s answer type directly by its interrogative (who, where, when), or by semantic mapping of other words in question (e.g. how much, what city, which year, etc.). The semantic class of the answer type is listed in table 3.1. Cooperating with the named entity finder, it does much help to locate and score answer in our QA system.

Answer Type	*Question Type*	*Example*
PERSON	who/what-who/ which-who	Hugo Young
LOCATION	where/what-where/ which-where	China
ORG	who/what-who/ which-who	Phoenix Suns
MONEY	how much/ how many money	Pounds 12m
PERCENTAGE	how much/ what-percentage	0.10%
DATE	when/what-date/ which-date	10 Feb 1994
TIME	what time	6:33 a.m.
DURATION	how long	9 1/2-month
LENGTH	how long	147 feet
SIZE	how large	1.5 million acres
NUMBER	how many	562

Table 3.1: Answer types of question

Not all questions can provide obvious clue of their goals. Some questions, which start from *what* and *which*, are ambiguous and scarcely say anything about specific answer type. We solve it by defining a concept named *question focus*.

Question focus can be interpreted as the most important part of question, which distinguishes the question from others at the most. It may be a word or a sequence of words. Low frequency word and proper noun/phrase is commonly chosen as question focus. For example, in the question "*What culture developed the idea of potlatch?*", the question focus is *potlatch*. Question focus makes it easier to filter the irrelevant document and locate the exact answer.

The *syntax pattern* of question is generated by question parser. The purpose of parsing is to predict the possible syntax structure of answer sentence. Take the question "*What is a caldera?*" for an example, the possible answer sentence may be like "*Caldera is ...* ", "*Caldera, ...* " or "*... known as Caldera.*".

Finally, the question analyzer produces a set of query term and sends them to the search engine. Each term of the query comprises three fields:

a. *Query Term*, word or phrase extracted from the question.
b. *Term Rank*, calculated by the term's syntax role in question and the word frequency.
c. *Search Mode*, the suitable searching method for this query term.

3.3 Candidate Window Searcher

Among the large document collection, we try to find some segments of information that may be relevant to the question in order to restrict the scope for further processing. In this phase, we search the entire corpus for the query and generate N best candidate windows, from which we will extract answer to each question.

Our search engine makes use of the Boolean retrieval model, which is modified to suit for the QA task. Firstly, we define four kinds of search modes, named "Single Word Search", "Common NP Search", "Proper NP Search" and "Quoted Part Search" respectively.

Single Word Search is used to search the query term, which has only one word. It aims at finding all the occurrences of the word in the corpus.

Common NP Search is just like operator "OR" in Boolean Information Retrieval in that the words being searched need not co-occur with each other. It is often used to search people's name, which often occurs partially in the corpus.

Proper NP Search is somewhat like operator "OR" in Boolean Information Retrieval, except it discards sentences that only contain familiar query words that can be found in dictionary. Therefore, the remaining sentences just contain OOD query words, such as named entity. For example, for the query term "*Star Trek*", the search engine will only retrieve sentences that contain the word "*Trek*". And the sentence contains both of the words "*Star Trek*" will be ranked higher than that contains "*Trek*" only. It makes intuitive sense that the word "*Star*" occurs too often in the corpus to depict the information need.

Quoted Part Search performs just like operator "EXACT MATCH" in Boolean Information Retrieval. It is used to search quoted name, such as name of films or books. It not only requires query words to co-occur, but the order of query words to be matched exactly in the corpus as well.

Then, we create N-best window ranked by their window scores.

For every matched sentence among the corpus, we scan forward and backward within the same articles to get a candidate window with the size of no more than 4k bytes. That is, we try to locate all 4k-byte windows containing one or more matched sentences. However, these matched sentences are included in only one of those candidate windows, no overlap is allowed.

The windows are scored by the formula given below:

$$WS_i = \sum_{t=1}^{k} \left(\frac{2 * mc_t}{c_t} + \frac{msc_i}{sc_i} \right) * w_{qs}^t \qquad (3.1)$$

Where, mc_t is the number of terms in each query that locate in the window, while c_t is the total number of terms in each query. msc_i is the number of total matched sentences in the window, while sc_i is the total number of sentences containing in the window. The former factor indicates the coverage of the query terms for the candidate window, whereas the latter favors candidate window with more occurrences of query terms. k is the number of query term for the question. w^i_{qs} is the weight of the ith query term. We assign weights to each query term according to its search mode and rank.

Sort all the windows by its score and select top N best window for further processing. In our experiments, we use 2k windows for every question at most.

3.4 Answer Extractor

The *Answer Extractor* identifies and extracts answers from the candidate windows. Each candidate window first passes through a named entity finder, which identifies names of person, location and organization, monetary units, dates, time, etc. By use of the answer type and question focus, all possible answers are located within the candidate window. For each possible answer, a 250-byte-long section in the candidate window named *answer-window* is then created. We evaluate each answer-window using the following four scores:

1) Matched_queries-score: Compute a match-score for each query term and sum them all. The match-score of query term is determined by its search mode, the match degree and the distance to answer-window of each match situation in the candidate window.
2) Query_coverage-score: Assign a coefficient to each matched query term in the answer window and cumulate them.
3) Syntax_pattern-score: If certain sentence in answer-window satisfies any predictive syntax pattern of the question, a correspondent score will be assigned to it.
4) Consistent_question_part-score: If certain part of question is found consistent in the answer-window, a score determined by the number of words in that part will be computed. It's a useful feature especially for back-formulation questions or coincident back-formulation situation.

The final score for a given answer-window is computed as:

$$\begin{aligned} \textit{final-score} = {} & k_1 * \textit{matched\_queries-score} + \\ & k_2 * \textit{query\_coverage-score} + \\ & k_3 * \textit{syntax\_pattern-score} + \\ & k_4 * \textit{consistent\_question\_part-score} \end{aligned}$$

where, the weight vector (k_1, k_2, k_3, k_4) depends on the question feature, table 3.2 shows our empirical weight vector values:

Question feature	(k_1, k_2, k_3, k_4)
Num of query term=1	(8,8,16,4)
Num of low-freq word=0	(8,4,16,12)
Otherwise	(8,8,8,8)

Table3.2. Weight vector for final-score of answer-window

3.5 Evaluation

In this section, we describe the performance of our system. The system is evaluated by the Mean Reciprocal Answer Rank (MRAR):

$$MRAR = \frac{1}{n}\sum_{i=1}^{n}(1/rank_i)\,. \qquad (3.2)$$

We submitted two runs in the 50-byte category and two runs in the 250-byte category. The first two runs are generated by using the top 100 candidate windows. The next two runs are by processing only top 24 candidate windows. The strict evaluation results are presented in Table 3.3.

The accuracy (measured by the percentage of questions correct) of our system fluctuates on various answer type. It is pleasant on questions demanding for PERSON (58%) and LOCATION (55%), but disappointing on DATE (35%) and NUMBER (25%). It is mainly because we concentrated on training the statistical model and worked little on rule-based identification, which is relatively simple but more useful on number-relevant named entity.

Run	*Category*	*Number / Percentage of questions correct*	*MRAR*
FDUT9QS1	50-byte (1)	200 / 29%	0.192
FDUT9QL1	250-byte (1)	313 / 46%	0.339
FDUT9QS2	50-byte (2)	187 / 27%	0.195
FDUT9QL2	250-byte (2)	288 / 42%	0.319

Table3.3. Performance in TREC-9

On the training corpus of TREC-8, our system did best while using top 24 candidate windows. But in TREC-9, the 250-byte run using top 100 candidate windows (FDUT9QL1) does better than that of top 24 one (FDUT9QL2). We presume it is caused by the variation on question style. The question of TREC-9 is shorter than that of TREC-8 on average. And some new question structure is too unfamiliar for us.

4. Discussion and Future Work

It is our first time to take part in TREC. Attending TREC-9 provides us further understanding of NLP technology. We have accumulated such knowledge resources as bilingual dictionary and Chinese synonym dictionary. We have also designed several NLP tools during this period, such as named entity finder, query translator, parser and search engine.

Although moderate performance has been achieved in our three systems, we still have a lot of things to do in the future. First, we need to enrich our knowledge resources, especially in English. We need to acquire knowledge from different domains and employ a comprehensive machine dictionary (e.g. WORDNET or HowNet) for semantic analysis. Currently, our three systems are developed almost independently. Next time, we will try to implement techniques developed for one system to another. For example, *feature selection* of filtering system can also play important role in the search engine. And *relevance feedback* in search engine is quite similar to adaptation in filtering.

Finally, we hope to apply the ideas and notions learned from TREC to corresponding tasks of our native language.

ACKNOWLEDGMENTS

This research was partly supported by NSF of China under contracts of 69873011 and 69935010, and 863 High Technology Project of China under contract of 863-306-ZD-02-02-4. We are thankful to Yaqian Zhou, Kaijiang Chen, Li Lian and Wei Qian for their help in the implementation of corpus and topic processing, syntactic parser and HMM based English named entity finder.

Reference

1. Kenney Ng. A maximum likelihood ratio information retrieval model. In Proceedings of the 8th Text Retrieval Conference TREC-8,1999
2. Wu Li-de, et. al, Large Scale Chinese Text Processing, Fudan University Press, 1997
3. C. Buckley, G. Salton, J. Allan, Automatic Retrieval With Locality Information Using SMART, Proceedings of the 1st Text REtrieval Conference (TREC-1), NIST Special Publication 500-207, 1992
4. Fabrizio Sebastiani, *Machine Learning in Automated Text Categorization*, Technical Report B4-31, Istituto di Elaborazione dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa, IT, 1999
5. Dan Moldovan, Sanda Harabagiu, et. al, LASSO: A Tool for Surfing the Answer Net. In Proceedings of the Eighth Text Retrieval Conference, 1999
6. D.M. Bikel, S. Miller, R. Schwartz and R. Weischedel, *Nymble: a High-Performance Learning Name-finder*. Proceedings of the Fifth Conference on Applied Natural Language Processing, Association for Computational Linguistics, pp. 194-201, 1997

Fujitsu Laboratories TREC9 Report

Isao Namba

Computer System Laboratory Fujitsu Laboratories Ltd.
{namba}@jp.fujitsu.com

Abstract

This year a Fujitsu Laboratory team participated in web tracks. For TREC9 we experimented passage retrieval which is expected to be effective for Web pages which contain more than one topic. To split document into passages, we used NLP based paragrah detecting program, not by fixed (variable) window size. But it did not produce better result for TREC9 Web data. For indexing large web data faster, we developped two techiniques. One is multi-partional selective sorting for inversion which is about 10-30% faster than normal quick sorting in sorting term-number, text-number pair. The other is compressed trie dictionary based stemming.

1 System Description

Except reranking by passage retrieval, and passage segementing program for index preprocessing, the frame work we used, is same as that of TREC8[1].

1.0.1 Teraß

Teraß[2] is a fulltext search library, designed to provide an adequate number of efficient functions for commercial service, and to provide parameter combination testing and easy extension for experiments in IR. For TREC9 we added functions for run time passage retrieval

1.0.2 trec_exec

trec_exec is designed for automatic processing of TREC. It contains a procedure controller, evaluation module , logging module, and all non-searching units such as query generation, query expansion and so on. trec_exec can execute all the TREC processing for one run in a few minutes, and it can be used for system tuning by hill-climing. But it was difficult to tune parameter control for TREC9 web data, because document set and queries for TREC9 is different from past trec data.

2 Common Processing

2.1 Indexing/Query Processing

2.1.1 indexing vocabulary

The indexing vocabulary consists of character strings made up of letters, numbers, and symbols, and no stop words were used in indexing. For TREC8, we modified the grammar of the token recognizer to accept acronyms with symbols such as U.S., and AT&T as one token.

2.1.2 Stemmer

As the experiment in TREC8[1] shows, SMART[3] stemmer seems to be stable, we used SMART.

2.1.3 Information in inverted file

Text number, term frequency, and term position are stored for the ad hoc task, and small web track for run time phrase processing and reranking by bigram extraction.

For experiment of passage retrieval, the delimiters of passage were also indexed.

2.1.4 Stop word list for query processing

As in the TREC8[1], we used a stop word list of about 400 words of Fox[4], and words with a high df (more than 1/7 of the number of all documents) were also treated as stop words.

2.1.5 Stop pattern removal

The expression of TREC queries are artificial, so frequently appearing patterns such as "relevant document" are stop patterns. We generalized this observation, and removed the words which meet one of the following condition.

1. Word in stopword list is a stopword.
2. Word which is not a proper noun[1], and whose df in TREC1-7 queries is more than 400*0.1 is a stop word.
3. Word bi-gram whose df in TREC1-7 queries is more than 400*0.02 is a stop pattern.
4. Word tri-gram whose df in TREC1-7 queries is more than 400*0.01 is a stop pattern.
5. All the words in a sentence that contains "not relevant" are stop words.
6. 4 words following "other than" are stop words.
7. 4 words following "apart from" are stop words.

2.2 Weighting Scheme

The term weight is $qtf * tf * idf$, and the score for one document is the sum of the term weights with co-occurence boosting.

1. qtf

   qtf is the combination of the following parameters

   $qtf = \sum_f fw * tf * ttw$

   where

   f is the topic field (title, description or narrative).

   fw is weight of the topic field. We set the value for the title field to 3.0, the value for the description field 1.5, the value for the narrative is 0.9. Some teams [5], [6],[7] used weighting depending on field type, and we take the same approach.

   tf is the bare frequency in each field.

   ttw is the term type weight. It is set to 3 for terms, and set to 1 for phrase(word bi-gram).

2. tf

   We simply used the tf part of OKAPI[5].

   $tf = \frac{(k_1+1)*term\_freq}{(k_1((1-b)+\frac{b*doc\_length\_in\_byte}{average\_doc\_length\_in\_byte})}$

   $k_1 = 1.5, b = 0.75$

3. idf

   We used a modified idf of OKAPI. We introduced a cut off point for low df words, and decreased the idf value for high df words.

   $idf = log_2 \frac{N-(n*\alpha)}{n}$
   N is the number of documents
   n is df if (df > 1/10000 * N) else
   $n = 1/10000 * N$
   α is set to 3

2.3 Co-occurence Boosting

As in TREC8, we use co-occurence boosting techinique which favours co-occurence of query terms in a document. Co-ocurrence boosting is implemented by simply multipling the boost ratio to the similarity of each term.

$$S_i = \sum_t B * W_{t,i}$$

S_i is the degree of similarity between a document and topics.
i is the document number.
t is a term that document$_i$ includes.
$W_{t,i}$ is the part of similarity of term$_t$ in document$_i$.
B is the boost-ratio by term co-occurrence.

The best parameter B depends on the query, but it is difficult to tune them for each query. So we set the B to 1.10 for the title word, to 1.05 for the description word, and to 1.03 for the narrative word, and to 1.0 for the word added by query expansion.

2.4 phrase(bi-gram)

Instead of traditional IR phrase (two adjacent non-stopword pair with order or without order), we permitted limited distance in phrase. The motivation for introducing fixed distance is that that non-stopword may exist between two adjacent words in a query, and it producued slightly better result in the past experiment.[1] The term weight of bi-gram is fixed as 1/3 of a single word, and the distance is set to 4.

2.5 Query Expansion

Query Expansion was used for the ad hoc task, and small web track. The Boughanem formula[5] was used to select terms.

---

[1] U.S appears 94 times in TREC1-7 queries.

$$TSV = (r/R - \alpha s/S).w^{(1)} \qquad (1)$$

$w^{(1)}$ is modified and more general version of Robertson/Sparck Jones weight.

The α was set 0.001, and k4 was -0.3, k5 was 1, and k6 was 64. The top 20 documents in the pilot search were supposed to be relevant, and the documents ranked from 500 to 1000 were supposed to be non-relevant. The top ranked 40 words which are not included in original query, which are not included in the stopword list of SMART, whose tsv score are more than 0.003, whose df are more than 60, and whose df are less than 200000 were added to the original query.

No collection enrichment technique was used.

2.6 Passage Retrieval

The average text size of TREC8 web data is large compared with past TREC collections. Its average text size is about 8KB. If the large web page contains more than one topic, scoring the page by its contents (large passage) not whole contents may produce better result. This requires techniques to split text by structure of topics. Using NLP techiniques developped for text summarization[8], we splitted the text into paragrahs, and indexed the text with topic boundary.

Following is the example of splitted text. The topic delimiter is <delim > tag, and the attribute "level" expresses level of paragraph. As the number of level becomes bigger, the size of paragrah becomes larger.

```
<document>
<DOCNO>WTX049-B01-2</DOCNO>
<BD>
<delim level=7/>
Table of Contents First-Time Startup
Overview of the First-Time Startup Process Default
Values Using the Setup Command Facility
Help Text Using the Setup Command Facility Power-
ing Up Your System Verifying Installed Software and
Hardware
Configuring Global and Interface Parameters Storing
the Configuration in Nonvolatile Memory
Sample Configuration
<delim level=1/>
This chapter includes sample worksheets filled in to
show you how this information is used when the setup
command  facility runs through the System Config-
uration Dialog.
Note Some configuration parameters discussed in this
document (and shown on the configuration work-
sheets) apply
only to routers that have the protocol translation op-
tion. If your router does not have protocol transla-
tion,
the interactive setup command facility does not
prompt you for these parameters.
<delim level=1/>
Overview of the First-Time Startup Process
The first time you start up the system, the setup
command facility operates automatically. An inter-
active
dialog called the System Configuration Dialog ap-
pears on the system console screen. The dialog nav-
igates you
through the configuration process by prompting you.
for the information you have recorded on the config-
uration
worksheets. The setup command facility also pro-
vides default values and help text for the configura-
tion parameters,
as described later in this section.
The setup command facility detects which interfaces
are installed and prompts you for configuration in-
formation
for each installed interface. When you finish con-
figuring one interface, the setup command software
prompts you
for the next interface and continues until they are all
configured.
At first-time startup, you must do the following:
Power up your router and if necessary, test for prob-
lems with system memory and CPU.
Verify software version and installed hardware and
software options.
Configure global parameters.
Configure interface parameters.
<delim level=2/>
:
:
Copyright 1988-1995
Cisco Systems Inc.
</BD>
</document>
```

We simply apply Okapi scoring (variation we used) to the passage, and merged fulltext scoring, and passage scoring. In the training by TREC8 web data, we set passage boundary level to 3, in that case average passage size was about 250 words. Merging his technique prodocues slightly better (1

BEST COPY AVAILABLE

point in average precision) result for TREC8 web track data, but did not result in improvment in TREC9 web track data whose average text size is about 4KB

3 Small Web Track official Runs

Four runs are submitted, Flab9atN, Flab9atdN, Flab9atd2N, and Flab9atdnN. In the Run id, the infix 'a ' means automatic, 't' means using title field, 'd' means using description field, and 'n' means using narrative field.

Name	Flab9t	Flab9tdN	Flab9td2N	Flab9tdnN
field	T	TD	TD	TDN
link	NO	NO	NO	NO
Average Prec	.136	.181	.187	.192
R-Prec	.153	.207	.208	.223
P@20	.157	.232	.226	.252
Retrieved	50000	50000	50000	50000
Rel-ret	2617	2617	2617	2617
Relevant	1179	1526	1490	1567
best/ >= med	2/25	0/31	0/31	0/35

Table 1: Official web track result

4 Speed up of indexing

Generally sorting based inversion takes these 4 steps.

1. STEP1 Apply stemmer to input text.
2. STEP2 Convert stemmed word to term-id. In most cases term-id to stemmed word is assigned by sequential order. Using hash may be fastest.
3. STEP3 Put [term-id,text-number,(offset)] pair(tuple) to work area
4. STEP4 If work area is full, then sort the area by ascending order of term-id,text-number,offset.

In sorting based inversion, stemming(+hashing) and sorting takes 70% of whole processing speed[1]. So speed up of above process leads to speed of whole processing.

4.1 Multi-paritional selective sorting

The fastest sorting algorithm is generally quick sort algorithm. But in sorting the pair of inverted file entry, we can expect the distribution of primary key(term_id). Because the word with high document frequency gets the smaller number, and the word with low document frequency gets the bigger number, they distribute in a log regression manner. Using this statistics, we can partion the sorting area into multi blocks at one time instead of partioning the sorting area in binary block (quick sort). Multi block partioning soring is faster than binary partioning sorting if partitioning is successful. [2]

The other techniques we introduced is using radic sort. The order of radic sort is $O(n)$, so it is expected to be faster than quick sort $On(log_2(n))$. But in practice, it is slower than quick sort for large data. It is because radic sort requires two buffers, and once copying between two buffers requires real memory access, it serverly slow down. But if sort target is small enough [3], it is surely faster than quick sort.

Using quick sort for large block, and radic sort for small block, we can improve the sorting speed of overall for inverted file entry.

The multi-paritional selective sorting algorithm is as follows.

1. Input is [term_id,text_number]
2. Prepare $n+1$ blocks. n is $log_2(maxterm\_id)$
3. Parition entry into $n+1$ blocks. The partitioning function is to put entry to $log_2(term_id)$th block.
4. Foreach blocks, apply sorting.
    (a) If partion is larger than 3/4 of L2 cache, use quick sort.
    (b) If partion is small than 3/4 of L2 cache, use radic sort.

The performace of this approach is depending on the partioning. In the experiment for VLC100, it is 10% faster than normal quick sorting, for WT10g, it is 30% faster than normal quick sorting. [4]

---

[2] multi partioning itself requires more complicated decision function than binary blocking sorting. So if partioning results in unbalanced blocks, it is slower than quick sort.

[3] it depends on the L2 cache of Hardware

[4] The speed depends on the max text_number in sorting target, and target area size etc.

4.2 Speed up of stemmer

As the stemming program matches rules step by step, it is slower than simple token recognizer. For example just recognizing token can process 16.5GB documents per hour, but with SMART stemmer it slows down 6.4GB documents per hour. Making compressed trie dictionary of frequently appearing 70000 words in text, and skip running stemming algorithm for them, the speed of stemmer increases about 13%.

5 Large Web Track

Our main concern this year is still how much resources are required to processing large data.

We concentrated on speed up of indexings.

5.1 Hardware environment

One PC was used for the large web track. It has 120GB disk, 640MB main memory and two Intel Celeron 466MHz CPUs. Its cost is about 180000 Japanse yen (about 1700 US$) at Nov 1999.

Using PC for information retrieval has practical advantages.

One is that PC(Intel i86) is cheaper than workstation. This is well known. The other is that the speed of information retrieval process is depending on the performance of integer calculation.[5] The PC(i86)'s processing speed of floating point calculation is slower than that of workstation, even its clock is 3 times faster than workstation, but this disadvantage is not critical in IR application.

5.2 language type checking

To reduce index size, and increase the speed of searching, statistical based language type checker is used as in TREC8[1]. Its effect is that the sum of word entry in inverted file is reduced to 10 million from 20 million, and the index size is reduced to 4.0GB from 4.8GB without stopword condition.

5.3 Large web track result

Our main concern is balancing processing speed and hardware cost. The submitted 3 runs are the same condition as that of TREC8. All runs did not use phrases, and query expansion. B+R means ranking document with AND condition of every nonstopword in a query. If the number of retrieved documents is less than 20, then ranking search is retried. This AND conditional interface is popular in actual Internet services. R means traditional accumulator method. Flab9bsN used index with stopwords. Table2 shows our official result.

Run-id	P@5	P@10	Calc	speed(sec)
Flab9bN	0.44	0.46	B+R	0.31
Flab9rN	0.44	0.45	R	0.72
FLab9bsN	0.45	0.45	B+R	0.47

Table 2: Large web official result

There is no remarkable difference in precision. B+R search and index with stopwords seems to be the best choice considering speed.

5.4 Performance of pre-processing

Compared with TREC8, we improved preprocessing speed. The preprocessing involves web detagging, running language type checking, and indexing. The official pre-processing data is as follows.

1. The detagging script for TREC8[1] is rewritten in C, and its processing time is improved to 10 hours including the time for gunzip the data.

2. language type checker takes 4 hours using 2 CPU.

3. Indexing

   Instead of Solaris2.6, we used Linux to avoid work memory swapping out problem. [1] The indexing time is 10 hours and 6 minutes with stopword condition.

condition	time	status	work area
With stopword	10.13 hours	official	300MB
Without stopword	12.15 hours	official	300MB

Table 3: Inversion time

The Index size is given in table 4.

---

[5] Ranking requires floating point calculation, but the most of CPU time is used for logical operation and decoding of inverted file entry.

files	with stopword	without stopword
inverted file	3.01	4.03
dictionary	0.46	0.59
text size array	0.07	0.07
text number id	0.41	0.41
total	3.95GB	5.10GB

Table 4: Index size

5.5 Performance of query processing

5.5.1 Average Processing Speed

The regulation of a large web track says that query processing speed is the total processing time divided by the number of query, As the experiment TREC8[1] shows, using two processs in 2 CPU environments is fastest. In TREC9 we used thread based approach. [6] The query processing speed is given in official result.

6 Conclusion

For small web track, we tried applying passage based scoring, but we did not get improvment. For large web track, we used two techniques, both of which are effective for speed up indexing.

References

[1] I Namba and N Igata. Fujitsu laboratories trec8 report. *The Eighth Text REtrieval Conference*, 2000.

[2] I Namba, N Igata, H Horai, K Nitta, and K Matsui. Fujitsu laboratories trec7 report. *The Seventh Text REtrieval Conference*, 1999.

[3] SMART ftp cite. ftp://ftp.cs.cornell.edu/pub/smart/. 1999.

[4] Chiristopher Fox. Chapter 7, lexical analysis and stoplists. *Information Retrieval Data Structure and Algorithms ed. William B. Frakes, Ricardo Baeza-Yates Prentice Hall*, 1992.

[5] S E Robertson, S Walker, and M Beaulieu. Okapi at trec-7. *The Seventh Text REtrieval Conference*, 1999.

[6] D R H Miller, T Leek, and R M Schwarts. Bbn at trec-7. *The Seventh Text REtrieval Conference*, 1999.

[7] James Allan, Jamie Callan, Mark Sanderson, Jinxi Xu, and Steven Wegmann. Inquery and trec-7. *The Seventh Text REtrieval Conference*, 1999.

[8] Yoshio Nakao. Summary zation by structure of topics(in japanese). *IPSJ Natural Lanuage Processing Group 132th meeting*, 1999.

---

[6] Using multi-threading techniques, we tried more complicated processing in experiment, but we don't report the experiment in this paper.

Hummingbird's Fulcrum SearchServer at TREC-9

Stephen Tomlinson[1], Tom Blackwell
Hummingbird
Ottawa, Ontario, Canada

February 6, 2001

Abstract

Hummingbird submitted ranked result sets for the Main Web Task (10GB of web data) and Large Web Task (100GB) of the TREC-9 Web Track, and for Stage 2 of the TREC-9 Query Track (43 variations of 50 queries). SearchServer's Intuitive Searching produced the highest Precision@5 score (averaged over 50 web queries) of all Title-only runs submitted to the Main Web Task. SearchServer's approximate text searching and linguistic expansion each increased average precision for web queries by 5%. Enabling SearchServer's document length normalization increased average precision for web queries by 10-30% and for long queries by 100%. Squaring the importance of the inverse document frequency (relevance method 'V2:4') increased average precision in the query track by 5%. Blind query expansion decreased average precision of highly relevants for web queries by almost 15%; the same method was neutral when counting all relevants the same.

1 Introduction

Hummingbird's Fulcrum SearchServer kernel is an indexing, search and retrieval engine which runs on Windows and UNIX platforms. SearchServer, originally a product of Fulcrum Technologies, was acquired by Hummingbird in 1999. The SearchServer kernel is embedded in 5 Hummingbird products, including SearchServer, an application toolkit used for knowledge-intensive applications that require fast access to unstructured information.

The SearchServer kernel supports a variation of the Structured Query Language (SQL), called SearchSQL, which has extensions for text retrieval. Almost 200 document formats are supported, such as Word, WordPerfect, PDF and HTML. Many character sets and languages are supported, including the major European languages, Japanese, Korean, Greek and Arabic. SearchServer's Intuitive Searching algorithms were updated for version 4.0 which shipped in Fall 1999, and in subsequent releases of other products. The next major kernel release works in Unicode internally and supports many more languages [4].

2 System Description

All experiments were conducted on a single-cpu desktop system, OTWEBTREC, with a 600MHz Pentium III cpu, 512MB RAM, 186GB of external disk space on one e: partition, and running Windows NT 4.0 Service Pack 6.

For most official TREC runs, an experimental version of SearchServer 5.0 was used (a different experimental version was used for the Query Track runs in September than the Web Track runs in July and

---

[1] Core Technology, Research and Development, stephen.tomlinson@hummingbird.com

August). Commercial release SearchServer 4.0 was used for one Main Web Task run and one Query Track run.

3 Setup

We describe how SearchServer was used to handle the Main Web Task (10GB of web data) and Large Web Task (100GB) of the TREC-9 Web Track, and Stage 2 of the TREC-9 Query Track (1GB of news and government documents).

3.1 Data

The WT10g collection of the Main Web Task was distributed on 5 CDs. We copied the contents of each CD onto the OTWEBTREC e: drive (e:\data\wt10g\cd1 - e:\data\wt10g\cd5). The cd5\info subdirectory, containing supporting information not considered part of WT10g, was removed to ensure it wasn't indexed. The 5157 .gz files comprising WT10g were uncompressed. No further pre-processing was done on the data. Uncompressed, the 5157 files consist of 11,032,691,403 bytes (10.3GB), about 2MB each. Each file contains on average 328 "documents", for a total of 1,692,096 documents.

The WT100g collection of the Large Web Task was distributed on 2 DLT-4000 tapes. We copied the contents onto a "compressed NTFS" area of OTWEBTREC's e: drive (e:\data\compressed\wt100g). The BASE10 and BASE1 subsets are not considered part of WT100g and were stored elsewhere. We uncompressed the 50,023 files comprising WT100g from .gz format (and Windows NT internally recompressed them on the compressed NTFS drive), which took 5 hours. No further pre-processing was done on the data. Uncompressed, the 50,023 files consist of 107,828,665,842 bytes (100.4GB). Based on the change in bytes free on the drive, we estimate the files occupied about 58.8 billion bytes (54.8GB) on the compressed NTFS drive. Hence, NTFS compression saved about 46GB of space, poor compared to gzip compression (which saved 71GB), but still worthwhile. Each file contains on average 371 "documents", for a total of 18,571,671 documents. (For more information on the WT100g collection, see [3].)

Text Research Collection Volume 1, Revised March 1994, more commonly known as "TREC Disk 1", was used in Stage 2 of the Query Track, and consists of a single CD. We copied its contents to e:\data\TREC\Vol1. The various README files and the DTD directory were removed because they are not considered part of the collection. The 1265 .Z files comprising the collection were uncompressed. No further pre-processing was done on the data. Uncompressed, the 1265 files consist of 1,265,137,373 bytes (1.2GB), about 1MB each. Each file contains on average 404 "documents", for a total of 510,637 documents. (For more information on the TREC Disk 1 collection, see [10].)

3.2 Text Reader

To index and retrieve data, SearchServer requires the data to be in Fulcrum Technologies Document Format (FTDF). SearchServer includes "text readers" for converting most popular formats (e.g. Word, WordPerfect, HTML, PDF, Excel, PowerPoint, etc.) to FTDF. A special class of text readers, "expansion" text readers, can insert a row into a SearchServer table for each logical "document" inside a container, such as directory or library file. Users can also write their own text readers in C for expanding proprietary container formats and converting proprietary data formats to FTDF.

The library files of WT10g and WT100g consisted of several logical documents, each starting with a <DOC> tag and ending with a </DOC> tag. After the <DOC> tag, the unique id of the document, e.g. WTX104-B01-1, was included inside <DOCNO>..</DOCNO> tags. Other HTTP header information, such as the URL of the document, appeared inside <DOCHDR>..</DOCHDR> tags. The content of the web document started after the </DOCHDR> tag and ended at the already-mentioned </DOC> tag. Most document's content were HTML format because only documents with mime type "text/html" were included in the collections, but on the web, some servers mislabel binaries and other file types as text/html. We made no attempt to screen out such mislabeled documents.

We wrote a custom text reader called cTREC to handle expansion of the library files of the WT10g and WT100g collections and to make a few conversions to the HTML format.

In expansion mode (/E switch), cTREC scans the library file and for each logical document determines its start offset in the file (i.e. offset of <DOC> tag), its length in bytes (i.e., distance to </DOC> tag), and extracts its document id (from inside <DOCNO>..</DOCNO> tags). SearchServer is instructed to insert a row for each logical document. The filename column (FT_SFNAME) stores the library filename. The text reader column (FT_FLIST) includes the start offset and length for the logical document (e.g. cTREC/w/100000/30000). The document id column (controllable with the /d switch), contains the document id.

In web track format translation mode (/w switch), cTREC would insert control sequences around the header to turn off indexing (i.e. from <DOC> down to the </DOCHDR> tag was not indexed). Indexing was also turned off around HTML tags, except for the content of META NAME/HTTP-EQUIV="DESCRIPTION/KEYWORDS/SUBJECT/TITLE" tags. Some entities were converted: the ones listed in the DTDs for the TREC disks 1-5, e.g. &eacute; to e, and numeric entities, e.g. &#233; to e. Because we knew the queries were all English, we didn't try to take advantage of SearchServer's rich character support capabilities, such as accent-indexing and recognition of semantically equivalent forms of Unicode.

The library files of TREC Disk 1 also consisted of several logical documents delineated by <DOC>..</DOC> tags and identified by a <DOCNO>, so the cTREC /E switch also handled expansion of these files. When invoked without the /w or /E switch, cTREC assumes it is reading a document from TREC disks 1-5 and by default inserts control sequences to turn off indexing around all tags listed in the TREC disk 1-5 DTDs, and converts all entities listed in the DTDs. By default, cTREC also turns off indexing for data delineated by tags indicating keyword fields (namely IN, CO, G, GV, RE, MS, NS, DESCRIPT or SUBJECT tags) because the original TREC guidelines did not permit using those fields (a /k option exists for overriding this guideline). Some other tagged data is not indexed by default (nor with the /k option) because its content isn't considered helpful (for TREC Disk 1, data delineated by DOCNO, FIRST, SECOND, FILEID, NOTE, UNK, BYLINE, C, CODE-213, DOCID, NOTE, T2, T4, AUTHOR, DATE, SO, ADDRESS, AUTHOR and JOURNAL tags is not indexed by default; a longer list exists to cover the other disks). cTREC currently doesn't differentiate its tag handling by collection type; for example, the <G> tag is a keyword field in the Wall Street Journal documents, but not in the Federal Register documents, but cTREC treats it as a keyword field in both, a minor limitation. cTREC looks ahead at most 8000 bytes for an end tag when it encounters a tag indicating indexing should be turned off; if the end tag is not found, indexing is not turned off.

3.3 Indexing

WT10g was indexed in one table in most runs, created with the following SearchSQL statement:

```
CREATE SCHEMA WT10GW CREATE TABLE WT10GW
(DOCNO VARCHAR(256) 128) PERIODIC
BASEPATH 'E:\DATA' STOPFILE 'MYTREC.STP' APPROX_ZONES '32';
```

The APPROX_ZONES '32' parameter specifies that an approximate search index should be built on the external text column (32). The STOPFILE parameter specified a file containing a list of 101 stopwords to not index, including all letters and single-digit numbers. The PERIODIC parameter prevents immediate indexing of rows at insertion time. The BASEPATH parameter specified the directory from which relative filenames of insert statements would be applied. The DOCNO column was assigned number 128 and a maximum length of 256 characters.

After creating the table, we added the string "IND:x16384;b4096;" to the wt10gw.cfg file to ensure an obscure internal dictionary limit wouldn't be encountered at index-time. This step is not necessary as of SearchServer 5.0 Beta2.

Into this table, we just inserted one row, specifying the top directory of WT10g, with this Insert statement:

```
INSERT INTO WT10GW ( FT_SFNAME, FT_FLIST )
VALUES ( 'WT10G', 'cTREC/E/d=128:s!cTREC/w/@:s');
```

To index the table, we just executed this Validate Index statement:

```
VALIDATE INDEX WT10GW VALIDATE TABLE
TEMP_FILE_SIZE 2000000000 BUFFER 256000000;
```

The VALIDATE TABLE option of the VALIDATE INDEX statement causes SearchServer to review whether the contents of container rows, such as directory rows and library files, are correctly reflected in the table. In this particular case, SearchServer initially validated the directory row by inserting each of its sub-directories and files into the table. Then SearchServer validated each of those directory and library file rows in turn, etc. Validating library file rows invoked the cTREC text reader in expansion mode to insert a row for each logical document in the library file, including its document id.

After validating the table, SearchServer indexed the table, in this case using up to 256MB of memory for sorting (as per the BUFFER parameter) and using temporary sort files of up to 2GB (as per the TEMP_FILE_SIZE parameter), to produce a dictionary of the unique words and reference file with the locations of the word occurrences (mostly unused in our experiments because we did no proximity searches nor search term highlighting). By default, SearchServer stores the original words, not just the stems.

For one of our Main Web Task runs (hum9td4), we used commercial release SearchServer 4.0, which is limited to 2GB reference files. Hence for that run we indexed WT10g in 2 tables. The first table contained CD1, CD2, and the first 4 directories of CD5 (WTX097-WTX100). The second table contained CD3, CD4 and the last 4 directories of CD5 (WTX101-WTX104).

Because of various internal limits in the experimental SearchServer version used for WT100g, we indexed WT100g in 12 tables (more than proved to be necessary). No approximate search index was built; however, some Large Web Task runs re-used the approximate search index of WT10g as a spell-corrector.

For the Query Track, we indexed TREC Disk 1 three times, once filtering keywords as per the traditional TREC guidelines, a second time with keyword fields included in the index, and a third time using commercial release SearchServer 4.0 (including keywords).

4 Search Techniques

For the Main Web Task, the 50 "topics" were in a file called "topics.451-500". The topics were numbered from 451-500, and each contained a Title (which was an actual web query taken from a search engine log), a Description (NIST's interpretation of the query, with spelling and grammar errors fixed), and a Narrative (a more detailed set of guidelines for what a relevant document should or should not contain).

For the Large Web Task, the 10000 web queries were in a file called "queries_10000". Queries were numbered from 20001-30000. There was no separate Title, Description or Narrative, just the original web queries, one per line.

For Stage 2 of the Query Track, there were 43 separate files of 50 queries each (variants of TREC topics 51-100).

We modified the example stsample.c program included with SearchServer to parse the TREC topics files, construct and execute corresponding SearchSQL queries, fetch the top 1000 or top 20 rows, and write out the rows in the results format requested by NIST. (The modified stsample.c was called QueryToRankings.c.)

SELECT statements were issued with the SQLExecDirect api call. Fetches were done with SQLFetch (typically 1000 SQLFetch calls per query in the Main Web Task and Query Track, and 20 SQLFetch calls per query in the Large Web Task).

4.1 Intuitive Searching

All queries used SearchServer's Intuitive Searching, i.e. the IS_ABOUT predicate of SearchSQL, which accepts unstructured text. For example, for topic 451 of the Main Web Task, the Title was "What is a Bengals cat?". A corresponding SearchSQL query would be:

```
SELECT RELEVANCE('V2:3') AS REL, DOCNO
FROM WT10GW
WHERE FT_TEXT IS_ABOUT 'What is a Bengals cat?'
ORDER BY REL DESC;
```

This query would create a working table with the 2 columns named in the SELECT clause, a REL column containing the relevance value of the row for the query, and a DOCNO column containing the document's identifier. The ORDER BY clause specifies that the most relevant rows should be listed first. Typically a statement such as "SET MAX_SEARCH_ROWS 1000" was previously executed so that the working table would contain at most 1000 rows. In cases where the data was indexed in more than one table, the FROM clause would specify a UNION of the tables, e.g. SearchServer 4.0 queries contained "FROM WT10GW1 UNION WT10GW2".

Our QueryToRankings program removed a short list of words from the given topics before presenting them to SearchServer: "documents", "document", "items", "item", "relevant". This was originally done for internal TREC-5 experiments based on the TREC-5 topics frequently containing these words (e.g. "A relevant item will mention..."). It was found to make almost no difference to the scores whether these words were excluded or not, so we didn't bother to expand the list further for the TREC-9 Main Web Task. For the Large Web Task, in which most queries were known to be phrased as questions, we additionally removed the words "do", "does", "find", "how", "me", "show", "tell", "what" and "why".

4.1.1 Secondary Term Selection

The IS_ABOUT predicate by default expands each word to a "superterm" comprising all the linguistic variants of the term, e.g. "run" is added for "ran" (linguistic expansion can be disabled with the VECTOR_GENERATOR parameter). Some of these superterms may be subsequently discarded when searching the table (secondary term selection). For example, the RELEVANCE_METHOD setting has an optional document frequency parameter for discarding all terms which occur in more than a specified percentage of the rows (based on the most frequently occurring variant of the term). Secondary term selection improves performance and prevents highlighting of unimportant terms.

In the SearchServer 5.0 web track runs, we experimented with a different term selection approach based on an estimate of how many rows the terms would bring into the search and which involved a different formula for term importance which incorporated the vector length. In the SearchServer 4.0 runs, in which the experimental approach wasn't available, a document frequency cutoff of 15% was normally used, because that cutoff in past experiments didn't hurt quality, but significantly improved performance.

4.1.2 Statistical Relevance Ranking

To calculate a relevance value for a row of a table with respect to the vector of terms (actually, superterms) resulting from secondary term selection, the inverse document frequency of the term and the number of occurrences of the term in the row (term frequency) are determined from the index. The length of the row (based on the number of indexed characters in all columns of the row, which is typically dominated by the external document), is optionally incorporated, and the number of occurrences of the term in the vector is also used. The full details of synthesizing this information into a relevance value are proprietary, but draws from [7] (particularly the Okapi approach to term frequency dampening) and [9]. SearchServer's relevance values are always an integer in the range 0 to 1000.

SearchServer's RELEVANCE_METHOD setting can be used to optionally square the importance of the inverse document frequency (by choosing a RELEVANCE_METHOD of 'V2:4' instead of 'V2:3'). Experiments on past TREC ad hoc topics found that V2:4 often worked better than V2:3, but past TREC ad hoc topics didn't contain spelling errors, which could be over-emphasized when squaring the idfs.

SearchServer's RELEVANCE_DLEN_IMP parameter controls the importance of document length (scale of 0 to 1000) to the ranking. We found 200 worked best on TREC-8 small web experiments and used 200 for most submitted TREC-9 runs.

4.2 Approximate Text Searching

The Title-only queries of the Main Web Task and the queries of the Large Web Task were unedited web queries from search engine logs which appeared to sometimes contain spelling errors; for example, a query containing "vanila ice creem" probably meant to say "vanilla ice cream".

SearchServer's approximate text searching is based on edit distance, also known as the Levenshtein distance, which is the minimum number of insertions, deletions and/or replacements needed to transform one pattern into another. SearchServer's approximate text searching is fast for error ratios up to at least one-third, i.e. when the allowed edit distance is one-third the length of the search term. An excellent overview of approximate text searching techniques may be found in [6].

We experimented with using SearchServer's approximate text searching to fix some spelling errors. The first step was to look up the closest matches of each term in the web query by increasing edit distance and decreasing number of rows containing the term. For example, the SearchSQL to find the closest matches to "vanila" is

```
SELECT TERM, FT_DISTANCE(TERM) AS NUMERRORS,
  MAX(ROW_COUNT) AS NUMROWS
FROM SEARCH_TERMS
WHERE TABLE_NAME CONTAINS 'WT10GW'
 AND COLUMN_NAME CONTAINS 'FT_TEXT'
 AND TERM CONTAINS 'vanila' BEST_MATCHES 1000
GROUP BY TERM
ORDER BY NUMERRORS, NUMROWS DESC;
```

The results in the case of "vanila" were

```
('VANILA',  0,  8)
('VANILLA',  1,  2427)
('MANILA',  1,  1763)
('VANIA',  1,  47)
('VANITA',  1,  21)
('VAXILA',  1,  11)
('DANILA',  1,  10)
('VANINA',  1,  9)
('VANNILA',  1,  5)
...
```

We used the default error ratio of 34% for all approximate searches, e.g. 2 errors were allowed in the 6-letter vanila query. The WT10G collection contained 5,792,772 distinct words. SearchServer used an in-memory approximate search index to substantially reduce the time needed to find the close matches.

If the original term appeared in fewer than 10 rows, then the closest matches were added to the query until the sum of the row counts was 10 or more, with the following adjustments:

- On the first pass through the list of close matches, only matches with the same Soundex code [5] were considered. For example, "vanilla" has the same Soundex code as "vanila", but "manila" does not. If the Soundex matches didn't sum to 10 or more rows, then the closest non-Soundex matches were added until the sum was 10 or more rows. (Note: Soundex was implemented in the QueryToRankings program, not SearchServer.)

- If the row sum was still less than 10 after adding all close matches, the last character of the term was dropped, and the process repeated. For example, after "vanila", the next search would be for "vanil", then "vani", etc.

In the case of "vanila", the term occurred in just 8 rows, fewer than 10 , so the next term, "vanilla" was considered. It was a Soundex match, so it was added to the query, and adding its row count of 2427 exceeded the sum requirement, so the search for more close matches ended.

In the case of "creem", the heuristic didn't work because "creem" appeared in 43 rows, more than our arbitrary parameter of 10. In retrospect, we probably should have added the first Soundex match which occurred in more rows than the original term (if any), making an arbitrary parameter unnecessary. The closest match to "creem" is "creek" (17,521 rows), which Soundex would filter out. The next closest match is "cream" (10,692 rows), which is the term we wanted. Also, it may be better to just sort close matches by decreasing number of rows and not differentiate by edit distance. This change would have properly handled the case of "australai" (4 rows); our heuristic generated "australi" (99 rows, 1 difference), but probably the best match was "australia" (75,297 rows, 2 differences).

The truncation heuristic was an attempt to deal with words stuck together, e.g. "londonengland", but it wasn't very successful. A better solution may be to include phrases in the word list.

We kept the original terms in the query, e.g. the query "vanila ice cream" was changed to "vanila ice cream vanilla". A downside of this approach is that the ranking algorithm treats vanila and vanilla as separate terms, hence over-weighting documents which happen to contain both terms. If we integrated this approach into SearchServer, we could treat the terms as variants of one term, like we do for linguistic variations of the term.

4.3 Row Expansion

In past TRECs, "query expansion" was considered necessary to produce top results [11]. We experimented with using row expansion to indirectly expand the query in 2 of our Main Web Task submissions. The approach was built on top of SearchServer. An optimized version may be implemented inside SearchServer in a future release.

After running the initial query (possibly already expanded by approximate text searching in the Title-only case), we retrieved the top 1000 rows (unless SearchServer returned fewer, which sometimes happened for Title-only queries). For each of the top 5 rows, we asked SearchServer's Intuitive Searching to "find more rows like this" (we call these row expansion queries), retrieving the top 1000 rows. We then combined the relevance values from each of the 6 result sets, giving a weight of 5 to the original query results, and weights of 1 to each of the 5 row expansion results. We did not include any negative information. Mathematically, this approach works out to be similar to Rocchio expansion. (A detailed description of a good Rocchio feedback technique is in [1].)

We always used the same parameters in row expansion queries as were used in the initial query, e.g. the same document length importance. We used the top 5 rows because in experiments on the TREC-8 small web we found somewhere from 3 to 8 rows usually gave best results.

5 Results

The evaluation measures are explained in an appendix of the conference proceedings. Briefly: **Precision** is the percentage of retrieved documents which are relevant. **Precision@*n*** is the precision after *n* documents have been retrieved. **Average precision** for a topic is the average of the precision after each relevant document is retrieved (using zero as the precision for relevant documents which are not retrieved). **Recall** is the percentage of relevant documents which have been retrieved. **Interpolated precision** at a particular recall level for a topic is the maximum precision achieved for the topic at that or any higher recall level. For a set of topics, the measure is the average of the measure for each topic (i.e. all topics are weighted equally).

Below we present an analysis of our results, including results of several unofficial "diagnostic" runs. Table 1 summarizes the "official" web track runs we submitted for judging in August 2000:

Run	**hum9te**	**hum9tde**	**hum9td4**	**hum9tdn**	**hum9w1**	**hum9w2**	**hum9w3**
Task	Main	Main	Main	Main	Large	Large	Large
Topic Fields	T-only	T+D	T+D	T+D+N	web	web	web
SearchServer Ver.	5.0Tr1[2]	5.0Tr1	4.0	5.0Tr1	5.0Tr1	5.0Tr1	5.0Tr1
Approx. Search	Y	N	N	N	Y	N	Y
Row Expansion	Y	Y	N	N	N	N	N
Linguistic Exp.	Y	Y	Y	Y	Y	Y	N
REL...METHOD	V2:3	V2:3	V2:3:15	V2:4	V2:3	V2:3	V2:3
REL...DLEN_IMP	200	200	200	200	200	200	0
REL...AVG_DLEN	10000	10000	10000	10000	10000	10000	n/a
Exp'l Sec. Term Sel.	Y	Y	N	Y	Y	Y	Y

Table 1: Summary of Runs submitted for the TREC-9 Web Track

---

[2] Build 5.0.0.61 with experimental changes for TREC

5.1 Main Web Task

The Main Web Task was to run 50 queries against 10GB of web data and submit a list of the top-1000 ranked documents to NIST for judging.

Topics were broken into 3 categories: automatic runs which used only the original Excite web query (called Title-only runs), automatic runs which used some other part of the topic statement (called Full Topic runs), and manual runs. We did not submit any manual runs, just automatic runs.

NIST produced a "qrels" file: a list of documents judged to be highly relevant, relevant or not relevant for each topic. From these, the scores were calculated with Chris Buckley's *trec_eval* program, which counts all relevants the same, including highly relevants. To produce scores which just counted highly relevants as relevant, we ran *trec_eval* a 2nd time on a modified version of the qrels file which had the ordinary relevants filtered out, then multiplied by 50/46 (in 4 of the 50 topics, there were no highly relevants). Hence the scores focused on highly relevants are averaged over just 46 topics.

The medians were derived from the statistics provided in the draft conference proceedings at the conference, which counted all relevants the same. For the Title-only category, the medians are the 9th-highest score of the 18 groups, just counting the highest score from each group in each measure. For the Full Topic category, the median is the 10th-highest score of the 19 groups.

5.1.1 Title-only runs

Table 2 shows Title-only runs produced with the experimental SearchServer 5.0 in July 2000. The only differences between these runs were the relevance method (V2:3 or V2:4) and whether or not row expansion post-processing was applied. Run 2b was the official "hum9te" run, which had the highest Precision@5 score and highest interpolated precision at the 10%, 20%, 30% and 40% recall levels of the 40 submitted Title-only runs from 18 groups:

SearchServer run	**AvgP**	**P@5**	**P@10**	**P@20**	**Rec0**	**Rec30**	**AvgH**	**H@5**	**H0**
2a: V2:3[3]	0.1949	32.0%	26.2%	22.0%	0.5223	0.2696	0.1948	15.7%	0.3264
2b: V2:3 + exp	0.1970	32.4%	25.4%	21.5%	0.4802	0.2808	0.1703	13.5%	0.2732
2c: V2:4	0.1931	29.6%	25.0%	21.5%	0.5170	0.2674	0.2078	15.7%	0.3441
2d: V2:4 + exp	0.1909	29.6%	23.8%	21.1%	0.4756	0.2774	0.1778	14.8%	0.2618
Median (18 grps)	0.1464	21.6%	21.2%	17.4%	0.4015	0.1993	n/a	n/a	n/a

Table 2: Precision of Title-only runs

Glossary:
AvgP: Average Precision (defined above)
P@5, P@10, P@20: Precision after 5, 10 and 20 documents retrieved, respectively
Rec0, Rec30: Interpolated Precision at 0% and 30% Recall, respectively
AvgH: Average Precision just counting Highly Relevants as relevant
H@5: P@5 just counting Highly Relevants as relevant
H0: Rec0 just counting Highly Relevants as relevant

---

[3] Scores for diagnostic runs may differ slightly from those given in the notebook paper because, for this paper, ties in relevance values were broken according to SearchServer's ordering in the result list.

Impact of Relevance Method (compare 2a to 2c, or 2b to 2d): V2:3 was modestly better at finding relevant documents (columns AvgP through Rec30), but V2:4 was modestly better at finding highly relevant documents (columns AvgH through H0, except in 1 H0 case).

Impact of Row Expansion (compare 2a to 2b, or 2c to 2d): Our experimental row expansion post-processing made little difference for relevants, but hurt the scores when focusing on highly relevants. Perhaps the finding of past TRECs that query expansion is usually necessary for top results is not valid when just focusing on highly relevants. However, the results of many groups will have to be considered, not just ours.

To measure the impact of approximate text searching and linguistic expansion, Table 3 shows runs which were done in January 2001 with a more recent SearchServer build (5.0.500.14). This version contained a new linguistic package and did not use the experimental secondary term selection (instead, terms in more than 15% of documents were discarded (relevance method V2:3:15)). No row expansion was done, and document length importance was increased to 750:

SearchServer Run	**AvgP**	**P@5**	**P@10**	**P@20**	**Rec0**	**Rec30**	**AvgH**	**H@5**	**H0**
3a: ling only	0.1919	30.4%	25.6%	21.5%	0.5265	0.2686	0.2343	15.7%	0.3548
3b: apx, ling	0.2019	32.0%	27.2%	22.9%	0.5516	0.2769	0.2509	16.5%	0.3647
3c: apx only	0.1914	32.4%	28.0%	22.8%	0.5586	0.2693	0.2273	17.0%	0.3898
3d: neither	0.1805	30.4%	26.4%	21.6%	0.5258	0.2562	0.2089	15.7%	0.3535

Table 3: Impact of Approximate Text Searching and Linguistic Expansion (Title-only runs)

Impact of Approximate Text Searching (compare 3a to 3b, or 3d to 3c): The spell-correction heuristics increased most precision scores by just 1-2 points. Almost all of the improvement was in 2 topics: 487 ("angioplast7", for which "angioplasty" was added) and 463 ("tartin", for which "tartan" was added). 2 topics became a little worse, 481 and 495, because "1920" was unnecessarily added for "1920's", apparently over-weighting that term.

Impact of Linguistic Expansion (compare 3c to 3b, or 3d to 3a): Linguistic expansion improved average precision, but slightly lowered Precision@10. In average precision, topic 469 was helped ("steinbach nutcracker") as was 490 ("motorcycle safety helmets"), but topic 492 was hurt ("us savings bonds") as was 458 ("fasting"). Note that all topics were English. SearchServer's linguistic expansion is likely to be more useful for languages with more noun forms, such as German and Finnish.

Table 4 shows additional runs just varying in the setting of document length importance (RELEVANCE_DLEN_IMP parameter). These runs used build 5.0.500.14, approximate text searching and linguistic expansion were both on, and the relevance method was 'V2:4:15':

DLen Importance	**AvgP**	**P@5**	**P@10**	**P@20**	**Rec0**	**Rec30**	**AvgH**	**H@5**	**H0**
4a: 0	0.1595	26.0%	21.2%	17.7%	0.4393	0.2222	0.1521	10.9%	0.2554
4b: 250	0.1908	29.6%	24.2%	21.2%	0.4960	0.2677	0.1926	14.8%	0.3084
4c: 500	0.2050	31.2%	26.0%	21.8%	0.5410	0.2828	0.2282	16.1%	0.3427
4d: 750	0.1992	30.0%	24.6%	21.7%	0.5539	0.2788	0.2528	16.1%	0.3878
4e: 1000	0.1744	27.6%	20.8%	18.7%	0.4892	0.2341	0.2350	16.1%	0.3318

Table 4: Impact of Document Length Normalization (Title-only runs)

Impact of Document Length Normalization: Ignoring document length (row 4a) hurt all scores; average precision was 10-30% higher in the other rows. The impact on highly relevants was even larger. Generally, we find that settings of 200 or higher all work pretty well for average precision, but higher settings appear to be better for finding highly relevants. The setting of 750 was probably the best overall in Table 4. See Table 6 for another document length experiment.

5.1.2 Full Topic runs

The other category for automatic runs were runs which included any part of the topic besides the Title field. The median precision scores were higher for this category, as were SearchServer's scores, which makes sense because the queries had the more detailed Description and/or Narrative fields included.

Table 5 shows Full Topic runs produced in July 2000. The differences between these runs were the relevance method (V2:3 or V2:4), whether or not row expansion post-processing was applied, whether or not commercial release SearchServer 4.0 was used, and whether or not the Narrative was included. Runs 5b, 5c and 5h were submitted (official runs "hum9tde", "hum9td4" and "hum9tdn" respectively). All runs were above the median in average precision:

SearchServer run	**AvgP**	**P@5**	**P@10**	**P@20**	**Rec0**	**Rec30**	**AvgH**	**H@5**	**H0**
5a: T+D, V2:3	0.2374	39.2%	30.8%	25.3%	0.6202	0.3336	0.2397	17.0%	0.3930
5b: 5a + exp	0.2217	37.2%	29.4%	24.0%	0.5217	0.3082	0.1783	14.8%	0.2722
5c: SS4, V2:3:15	0.2115	37.6%	30.8%	24.8%	0.5990	0.3051	0.2053	15.2%	0.3628
5d: 5a + Narr	0.2184	39.2%	34.0%	26.3%	0.6059	0.3201	0.2005	15.2%	0.3313
5e: T+D, V2:4	0.2347	36.0%	30.6%	25.2%	0.5617	0.3190	0.2372	17.8%	0.3635
5f: 5e + exp	0.2228	34.0%	28.6%	24.9%	0.4895	0.3114	0.1862	14.3%	0.2675
5g: SS4, V2:4:15	0.2380	36.0%	30.8%	25.0%	0.5735	0.3197	0.2301	17.4%	0.3659
5h: 5e + Narr	0.2335	42.0%	35.2%	27.4%	0.6391	0.3341	0.2158	16.5%	0.3790
Median (19 grps)	0.1948	36.0%	31.4%	24.6%	0.6029	0.2692	n/a	n/a	n/a

Table 5: Precision of Full Topic runs

Impact of Row Expansion (compare 5a to 5b, or 5e to 5f): Row expansion post-processing hurt all scores, especially for highly relevants, as in the Title-only case.

Impact of Relevance Method (compare 5a to 5e, 5b to 5f, 5c to 5g, or 5d to 5h): More often than not, V2:4 was a little better, including for highly relevants, but it made little difference.

Impact of including the Narrative (compare 5a to 5d, or 5e to 5h): Including the Narrative hurt average precision scores. It increased relevants early in the result list, but not highly relevants.

Difference from SearchServer 4.0 (compare 5a to 5c, or 5e to 5g): SearchServer 4.0, which split the data into 2 tables and used the simpler secondary term selection, produced scores which were a little lower than SearchServer 5.0's with V2:3, and about the same with V2:4.

Table 6 shows a dramatic result when re-doing the runs of Table 4 (RELEVANCE_DLEN_IMP experiment) with the Description and Narrative included:

DLen Importance	**AvgP**	**P@5**	**P@10**	**P@20**	**Rec0**	**Rec30**	**AvgH**	**H@5**	**H0**
6a: 0	0.1136	20.8%	18.4%	15.9%	0.3684	0.1672	0.1077	7.8%	0.1846
6b: 250	0.2233	40.0%	34.8%	28.0%	0.6320	0.3219	0.2007	16.5%	0.3628
6c: 500	0.2435	44.8%	35.2%	29.5%	0.6833	0.3330	0.2447	19.1%	0.4264
6d: 750	0.2569	43.2%	36.8%	30.1%	0.6958	0.3384	0.2843	20.9%	0.4845
6e: 1000	0.2454	42.0%	36.6%	28.0%	0.6894	0.3388	0.2908	20.9%	0.4825

Table 6: Impact of Document Length Normalization (T+D+N runs)

Impact of Document Length Normalization: Ignoring the document length (row 6a) significantly hurt all scores; average precision was about 100% higher in the other rows. Many irrelevant long documents (e.g. 500KB or more) were brought into the search result by the numerous, relatively unimportant terms in the long queries when there was no document length adjustment. It appears that even a low setting, such as 250, was enough to overcome the issue. As in the Title-only case, the impact was even larger for highly relevants, and higher settings were better. Again, probably 750 was the best setting overall in Table 6.

5.2 Large Web Task

Results of our Large Web Task runs are summarized in Table 7. Only 84 of the 10000 queries were judged, and only the top 10 documents submitted for each query were judged:

SearchServer Run	**Reciprocal Rank of First Satisfactory**	**Precision@1**	**Precision@5**	**Precision@10**
hum9w1	0.4381	30.95%	32.62%	32.50%
hum9w2	0.4262	30.95%	31.43%	31.67%
hum9w3	0.4174	28.57%	30.24%	29.40%

Table 7: Precision of Large Web Task runs

The use of approximate text searching for handling misspelled terms (used in hum9w1 but not hum9w2, the only difference) improved most precision scores by 1 point. The benefit primarily came from query 28616 ("where can i find a good deal on a mothermoard") for which the term "motherboard" was helpfully added.

Turning off document length normalization and linguistic expansion (as done in hum9w3, the only differences from hum9w1) lowered precision scores by just 2-3 points. This result is in line with what one would expect from the Main Web Title-only findings for Precision@5 and Precision@10, which suggest that removing the document length adjustment hurt 3-4 points, but disabling linguistics helped 0-1 points. Unfortunately, the pool of documents submitted per topic is too small for this task for us to run meaningful experiments on isolated factors after the fact like we could for the Main Web, e.g. perhaps the individual impact of document length and linguistics is actually higher in this task.

We divided WT100g into 12 tables, each with their own set of inverse document frequencies. This need not lower the scores: AT&T found that Precision@10 scores were actually a little higher when they split WT100g into 20 tables in TREC-8 (see [8]). However, our preliminary experiments with global idfs suggest that the table-splitting may have cost us several points of precision; e.g. with global idfs, we get 30% in precision@10 with 30% unjudged, and many of the unjudged appear to be satisfactory. Our experimental secondary term selection was set more aggressively for this task than in the Main Web, and there would have been some inconsistencies in the terms discarded for different tables in our official runs.

For query 27028 ("s3 patches"), we regret that 's' and '3' were stop words. Perhaps we should enable SearchServer's option of parsing numbers as if they were letters.

5.3 Query Track

The Query Track is for evaluating not just retrieval systems, but the effect of query variations on such systems. For background on this track, see [2].

In Stage 1 of the Query Track, participants created variations of old TREC topics 51-100, including very short queries (2-3 words), natural language sentence queries, and queries based on reading system results without consulting the original topics. In all, 43 sets of 50 queries were produced by 6 different groups. We did not submit any queries for Stage 1.

In Stage 2, all groups, including those which did not submit queries, were asked to run all the query sets on their systems. The more systems, the more reliable the conclusions about varying queries. We contributed 7 runs. The overall average precision scores for each of these runs (averaged over all 43*50 queries) are in Table 8:

Run	**AvgP**	**Experiment** (i.e. what was different from baseline)
humB*	0.1732	baseline
humK*	0.1713	keyword fields were not indexed (/k option of cTREC text reader was not used, see section 3.2)
humD*	0.1771[4]	document length importance was set low (RELEVANCE_DLEN_IMP was set to 200 (baseline was 750))
humV*	0.1648	inverse document frequency not squared (RELEVANCE_METHOD was 'V2:3:15' (baseline was 'V2:4:15'))
humA*	0.1741	approximate text searching added fixes for spelling errors (algorithm of section 4.2 except the table used to index TREC Disk 1 with keywords was used)
hum4*	0.1713	SearchServer 4.0 was used (baseline used experimental SS 5.0 which contained a new linguistic expansion package which was known to still have a few glitches)
humI*	0.1736	terms in more than 15% of rows not discarded (RELEVANCE_METHOD was 'V2:4:100' (baseline was 'V2:4:15'))

Table 8: Average Precision of Query Track runs

These results suggest that excluding keyword fields makes little difference. Decreasing the document length importance was modestly helpful. SearchServer's relevance method 'V2:4', which squared the importance of the inverse document frequency, produced modestly better results. The attempt at handling spelling errors was of only minor benefit, though probably relatively few queries contained spelling errors (compared to the web queries). The experimental SearchServer 5.0 scores were slightly higher than those of SearchServer 4.0, despite some known glitches in the new linguistic package, such as expanding "in" to "Indiana" (since ironed out).

Perhaps the most interesting result is that excluding terms which occur in more than 15% of the documents, which helps search-time performance, didn't decrease average precision significantly. This result is consistent with other experiments we have done, but differs from the finding reported in Managing Gigabytes that discarding frequent terms "greatly reduces retrieval effectiveness" [12] (page 427).

---

[4] We received corrections to the humD and humV results after submitting the notebook paper.

References

[1] Chris Buckley, Amit Singhal and Mandar Mitra. Using Query Zoning and Correlation Within SMART: TREC 5. In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*. NIST Special Publication 500-238. http://trec.nist.gov/pubs/trec5/t5_proceedings.html

[2] Chris Buckley and Janet Walz. The TREC-8 Query Track. In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*. NIST Special Publication 500-246. http://trec.nist.gov/pubs/trec8/t8_proceedings.html

[3] David Hawking, Nick Craswell and Paul Thistlewaite. Overview of the TREC-7 Very Large Collection Track. In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*. NIST Special Publication 500-242. http://trec.nist.gov/pubs/trec7/t7_proceedings.html

[4] Andrew Hodgson. Converting the Fulcrum Search Engine to Unicode. In *Sixteenth International Unicode Conference*, Amsterdam, The Netherlands, March 2000.

[5] Donald E. Knuth. The Art of Computer Programming, Vol. 3: Sorting & Searching, 2nd edition Revised, January 1998. Addison Wesley Longman.

[6] Gonzalo Navarro. Approximate Text Searching, PhD Thesis, Dept. of Computer Science, University of Chile, December 1998.

[7] S.E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford. (City University.) Okapi at TREC-3. In D.K. Harman, editor, *Overview of the Third Text REtrieval Conference (TREC-3)*. NIST Special Publication 500-226. http://trec.nist.gov/pubs/trec3/t3_proceedings.html

[8] Amit Singhal, Steve Abney, Michiel Bacchiani, Michael Collins, Donald Hindle and Fernando Pereira. AT&T at TREC-8. In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*. NIST Special Publication 500-246.

[9] Amit Singhal, John Choi, Donald Hindle, David Lewis and Fernando Pereira. AT&T at TREC-7. In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*. NIST Special Publication 500-242. http://trec.nist.gov/pubs/trec7/t7_proceedings.html

[10] Ellen M. Voorhees and Donna Harman. Overview of the Eighth Text REtrieval Conference (TREC-8). In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*. NIST Special Publication 500-246. http://trec.nist.gov/pubs/trec8/t8_proceedings.html

[11] Ellen M. Voorhees and Donna Harman. Overview of the Seventh Text REtrieval Conference (TREC-7). In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*. NIST Special Publication 500-242. http://trec.nist.gov/pubs/trec7/t7_proceedings.html

[12] Ian H. Witten, Alistair Moffat and Timothy C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. 2nd edition, 1999. Morgan Kaufmann Publishers.

Hummingbird, Fulcrum, SearchServer, SearchSQL and Intuitive Searching are the intellectual property of Hummingbird Ltd. All other company and product names are trademarks of their respective owners.

English-Chinese Information Retrieval at IBM

Martin Franz, J. Scott McCarley, Wei-Jing Zhu
IBM T.J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Abstract

We describe TREC-9 experiments with an IR system that incorporates statistical machine translation trained on sentence-aligned parallel corpora for both query translation (English⇒Chinese) and document translation (Chinese⇒English .) These systems are contrasted with monolingual Chinese retrieval and with query translation based on a widely available commercial machine translation package. These systems incorporate both words and characters as features for the retrieval. Comparisons with a baseline from TREC-5/6 enable our experiments to address issues related to the differences between Beijing and Hong Kong dialects.

1 Chinese preprocessing

The TREC-5/6 corpus is in the Taiwanese dialect of Chinese, and is encoded in the GB-2312 character set. The TREC-9 corpus consists of news stories from Hong Kong, and is encoded in the Big-5 character set. In order to perform comparable experiments on both corpora, we adopt UTF-8 encoded unicode as our internal representation of Chinese characters. In order to study baseline retrieval performance, we converted the TREC-5/6 Chinese track corpus from GB to unicode. We converted the TREC-9 corpus from Big-5 to unicode (ignoring the "extra" HKSAR hanzi.) We note that unicode often contains at different code points both the simplified and traditional forms of the same hanzi; the mappings relating the simplified and traditional forms, as well as other semantic variants within unicode are well-documented [1]. Any character that could be linked to a simplified Chinese character (including indirect linkings) was mapped to that character; simplified characters linked to each other were mapped to the smaller unicode number.

BEST COPY AVAILABLE

2 Chinese IR System Description

The Chinese IR track in TREC-5/6 triggered extensive experimentation on whether Chinese characters should be automatically tokenized ("segmented") into words to use as features for IR, or whether the characters themselves (and n-grams of characters) should be used as tokens for IR. No clear consensus has emerged [2, 3], see also [4, 5], although there are good reasons to prefer shorter words (limited to less than about 4 characters) [6] as well as to incorporate both types of features. Our approach to incorporating both words and characters is to build two separate systems, closely modeled on our English IR system [7], and to merge the results by linear combination of scores.

Both corpora were segmented with a statistical segmenter similar to the one discussed in [8]. The corpus-based iterative approach to Chinese segmentation allowed us to customize the segmenter's language model probabilities to each corpus. The segmenter's vocabulary consisted mostly of two-character words, with no words exceeding 5 characters, since there is evidence that short words are preferable (longer words often fail to match any terms in queries) for information retrieval purposes.

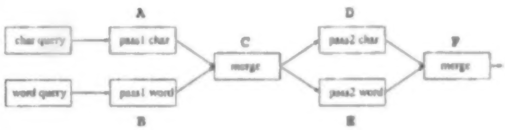


Figure 1: Diagram of system

Our Chinese (monolingual) IR system is a two-pass system, in which the results of the initial retrieval are used to construct an expanded query, which is then used for a second pass retrieval. The outline of the system is indicated in Fig. (1). For generality of explanation, we assume that the query has already been preprocessed into two forms, one in which it has been automatically been tokenized into short words for use as IR features, and one in which each character is a separate token. The first pass scoring is based on the Okapi formula [9], using the characters as features in (A), and using short-word tokens as features in (B). The results are merged at (C) by linear combination of scores. Our query expansion is based on LCA [10], which selects features from the top-ranked documents (output at (C)) which frequently cooccur with query features in these documents. At (D) we query-expand the character-based representation, i.e., we look for characters that frequently cooccur with query characters in the top-ranked documents. At (E) we query-expand the short-word-based representation of the corpus. Both query expansions are merged at (F) to yield the final results.

BEST COPY AVAILABLE

3 Crosslingual IR Experiments

3.1 Query translation with a statistical model

We used two parallel corpora (Hong Kong Laws and Hong Kong News, available from the Linguistic Data Consortium as part of the Topic Detection and Tracking (TDT) project [12]), and a smaller amount of material from the FBIS, to build a *character-based* statistical translation model. Because the majority of parallel text was from Hong Kong, we expect this translation model to be particularly well-matched to the TREC-9 test set, and less well suited for the TREC-5/6 baseline (in contrast to the commercial translation package described above.) We built a model of the probability $p(c|E, E_+, E_-)$ where c is a Chinese character, E is an English word, and E_+ and E_- are the nearest following and preceding content words. Models of this structure have previously been described in [13]. These models are trained from a sentence-aligned parallel corpus, together with a word alignment. The word alignment is constructed automatically from the parallel corpus using Poisson-fertility model, as described in [14, 15] This model predicts only characters, not the order of characters. The ordering was imposed when possible by dictionary lookup, using a Chinese-English dictionary made available for the TDT project. We note that ordering the characters was not necessary for the character-based aspect of IR, but was necessary in order to segment the query into words (and hence for the word-based aspect of IR.) Experiments with this model are denoted QT in the results.

3.2 Document Translation with a Statistical Model

Because of our prior success mixing query translation and document translation [16], we also built a Chinese⇒English translation model from the same parallel corpora as above. This model is not directly comparable to statistical query translation model - it is a word-based model for $p(E|C)$ the probability of an English (morphological root) word E given a Chinese word C (determined from an automatic segmentation of the corpus. This model is also a Poisson- fertility model. When the corpus was translated, the translation model was supplemented by the LDC dictionary. Since the resulting translation of the corpus is English, there is no character/word distinction in the IR system associated with this retrieval. Results with this model are denoted DT in the results.

3.3 Query translation using commercial software

Another set of experiments involved translating the English version of the query using a widely available commercial machine translatation package [11]. These experiments will be denoted TW in the results. Since this software was developed in Taiwan, we expected that it would be more closely matched to the TREC

5/6 baseline than to the the TREC-9 test set. The output of the translation package, which consists of unsegmented characters, is automatically segmented as into words and characters and then used in our Chinese IR system.

4 Discussion of Results

The character-based half of the system generally outperformed the word-based half of the system, across both types of Chinese⇒English translation, and monolingually, especially on the first pass of scoring. Query expansion made the differences between character- and word-based retrieval less clear. The gain from mixing character-based and word-based results was only slight. This result seems to be true for both the TREC-5/6 set and the TREC-9 set, and so is probably independent of dialect. On the other hand, dialect strongly influenced the relative behavior of the two query translation systems. The Taiwan-built commercial system, as expected, performed better on the TREC 5/6 task (Beijing data), whereas the statistical system, trained on Hong Kong data, performed better on the TREC-9 task (Hong Kong corpus.)

Our submission system was a merging of the TW, QT, and DT systems. However, the relative ranks of the TW, QT, and DT systems are completely reversed between TREC-5/6 and TREC-9, presumably mostly as a result of dialect differences. Thus TREC-5/6 was could not be used as a training set to predicting merging weights, etc. for TREC-9. However, in both sets (unlike many other IR tasks) the value of merging the results of different systems questionable.

5 Acknowledgements

This work is supported by DARPA under SPAWAR contract number N66001-99-2-8916. We would like to thank Salim Roukos and Todd Ward for valuable discussions.

References

[1] www.unicode.org/Public/MAPPINGS/EASTASIA/UNIHAN.TXT

[2] "Spanish and Chinese Document Retrieval in TREC-5", A.Smeaton and R. Wilkinson, in *Proceedings of the Fifth Text REtrieval Conference (TREC-5)* ed. by E.M. Voorhees and D.K. Harman. NIST Special Publication 500-238, 1997

[3] "Chinese Document Retrieval at TREC-6", R. Wilkinson, in *Proceedings of the Sixth Text REtrieval Conference (TREC-6)* ed. by E.M. Voorhees and D.K. Harman. NIST Special Publication 500-240, 1998

system	aveP TREC 5-6	aveP TREC 9
ML (c)	0.4783	0.2887
ML (w)	0.4813	0.3112
ML (c+w)	0.4904	0.2973
TW (c)	0.3152	0.1865
TW (w)	0.2689	0.2153
TW (c+w)	0.3193	0.2030
QT (c)	0.2778	0.2402
QT (w)	0.2228	0.1805
QT (c+w)	0.2810	0.2420
DT (w)	0.2889	0.2181
QT+DT	0.2979	0.2203
all 3	**0.3054**	**0.2258**

Table 1: Final scores by subsystem: ML = monolingual baseline, TW = query translation with commercial software QT = statistical query translation, DT = statistical document translation

[4] "On Chinese Text Retrieval" J.-Y. Nie, M. Brisebois, and X.Ren, in *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* Zurich, Switzerland, pp. 4-11.

[5] K.L. Kwok "Comparing Representations in Chinese Information Retrieval", in SIGIR 1997.

[6] K.L. Kwok "Lexicon Effects on Chinese Information Retrieval", in EMNLP, 1997

[7] M. Franz, J.S.McCarley, S.Roukos, "Ad hoc and Multilingual Information Retrieval at IBM", in *Proceedings of the Seventh Text REtrieval Conference (TREC-7)* ed. by E.M. Voorhees and D.K. Harman, 1999

[8] X.Luo and S. Roukos, "An Iterative Algorithm to Build Chinese Language Models", in *34th Annual Meeting of the Association for Computational Linguistics* Santa Cruz, CA, 1996.

[9] S.E. Robertson, S. Walker, S. Jones, M.M. Hancock-Beaulieu, M. Gatford, "Okapi at TREC-3" in *Proceedings of the Third Text REtrieval Conference (TREC-3)* ed. by D.K. Harman. NIST Special Publication 500-225, 1995.

[10] J. Xu and W. B. Croft 1996 Query Expansion Using Local and Global Document Analysis. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* Zurich, Switzerland, pp. 4-11.

[11] TransWhiz C-E Translation PRO Version 3.0 package, by American Insight, Inc. (39-02 Main Street, 3F, Flushing, NY 11354)

[12] http://morph.ldc.upenn.edu/Projects/Chinese/

[13] J.S. McCarley and S.Roukos, "Fast Document Translation for Cross-Language Information Retrieval", in *Machine Translation and the Information Soup* ed. by D.Farwell, L.Gerber, and E.Hovy. (1998)

[14] P. F. Brown et al. "The mathematics of statistical machine translation: Parameter estimation", *Computational Lingustics*, 19 (2), 263-311, June 1993.

[15] S. Della Pietra, M. Epstein, S. Roukos, T. Ward "Fertility Models for Statistical Natural Language Understanding" *35th Annual Meeting of the Association for Computational Linguistics* Madrid, Spain, 1996.

[16] J.S. McCarley, "Should we Translate the Documents or the Queries in Cross-Language Information Retrieval?", in *37th Annual Meeting of the Association for Computational Linguistics* College Park, MD, 1999.

IBM's Statistical Question Answering System

Abraham Ittycheriah, Martin Franz, Wei-Jing Zhu, Adwait Ratnaparkhi
P.O.Box 218,
Yorktown Heights, NY 10598
{abei,franzm,wjzhu,adwaitr}@watson.ibm.com

Richard J. Mammone
Dept. of Electrical Engineering, Rutgers University,
Piscataway, NJ 08854
mammone@caip.rutgers.edu

Abstract

We describe the IBM Statistical Question Answering for TREC-9 system in detail and look at several examples and errors. The system is an application of maximum entropy classification for question/answer type prediction and named entity marking. We describe our system for information retrieval which in the first step did document retrieval from a local encyclopedia, and in the second step performed an expansion of the query words and finally did passage retrieval from the TREC collection. We will also discuss the answer selection algorithm which determines the best sentence given both the question and the occurrence of a phrase belonging to the answer class desired by the question. Results at the 250 byte and 50 byte levels for the overall system as well as results on each subcomponent are presented.

1 System Description

Systems that perform question answering automatically by computer have been around for some time as described by (Green et al., 1963). Only recently though have systems been developed to handle huge databases and a slightly richer set of questions. The types of questions that can be dealt with today are restricted to be short answer fact based questions. In TREC-8, a number of sites participated in the first question-answering evaluation (Voorhees and Tice, 1999) and the best systems identified four major subcomponents:

- Question/Answer Type Classification
- Query expansion/Information Retrieval
- Named Entity Marking
- Answer Selection

Our system architecture for this year was built around these four major components as shown in Fig. 1. Here, the question is input and classified as asking for an answer whose category is one of the named entity classes to be described below. Additionally, the question is presented to the information retrieval (IR) engine for query expansion and document retrieval. This engine, given the query, looks at the database of documents and outputs the best documents or passages annotated with the named entities. The final stage is to select the exact answer, given the information about the answer class and the top scoring passages. Minimizing various distance metrics applied over phrases or windows of text results in the best scoring section that has a phrase belonging to answer class. This then represents the best scoring answer.

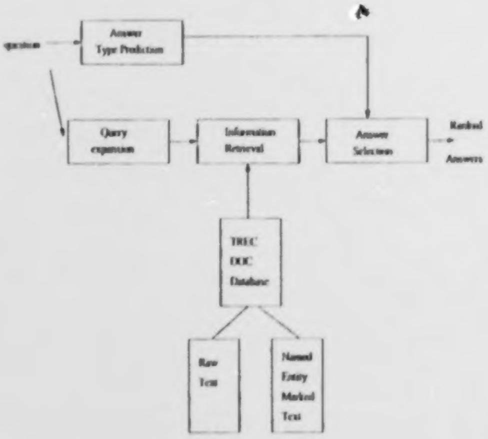


Figure 1: Question Answering Architecture

BEST COPY AVAILABLE

Maximum entropy modelling is described in (Della Pietra et al., 1995; Berger et al., 1996). Methods of feature selection is further described in (Berger and Printz, 1998). We will not discuss the mathematical details of the algorithm here, instead we will only show the features that are used in such a model.

We will now describe each sub-component in greater detail.

2 Answer Type Classification

In answer type classification the problem is to label a question with the label of the named entity that the question seeks. Our labels are the standard MUC (Chinchor, 1997) categories with the addition of PHRASE which is a catch all for answers not of the standard categories. In addition we had a REASON category which was tied to WHY questions. Processing of REASON and PHRASE is the same in our system, interpreting it as desiring a clause which had a noun phrase embedded in it.

A maximum entropy model of the process was trained on a corpus of questions that has been annontated with the above mentioned categories. We created 1900 questions by presenting a human subject a document selected at random and having read a portion of the document, a question was phrased, the answer and document number noted in addition. We also used 1400 questions from a trivia database (Hallmarks, 1999) annotated in a similar manner.

In the experiments we used the following types of features shown in Table 2. Each feature type expands on the feature above it. The "Expanded Hierarchy" feature uses WordNet (Miller, 1990) to expand words from a question word upto and including the first noun cluster. The "Mark Question Word" feature identifies the question words and labels them as occuring in the beginning of a question (bqw), in the middle (mqw) of a question or at the end of a question (eqw).

The features of the maximum entropy model were n-grams of words (required to be adjacent) and bag of words where position is not important. The performance of the algorithm is shown in Fig. 2. Each feature type adds to the accuracy of the system and choosing 700 features achieves the best accuracy (9.05%) on a held out subset of the data.

A peculiar feature of the architecture is that improvements in answer type prediction do not correlate directly with improvements in the overall score of the system. The reason is that parallel improvements must be made in the named entity marking as well as answer selection in order to realize them in the overall system.

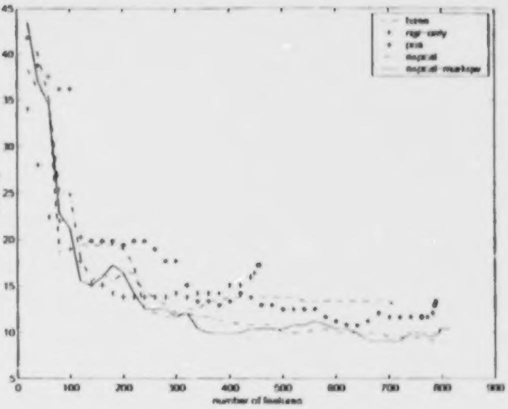


Figure 2: Answer Type Prediction Performance

3 Information Retrieval

The purpose of the information retrieval module is to search the database to select passages of text, containing information relevant to the query.

Our information retrieval subsystem uses a two-pass approach. In the first pass, we searched an encyclopedia database. The highest scoring passages were then used to create expanded queries, applied in the second pass scoring of the TREC passages. We used data pre-processing and relevance scoring techniques similar to the ones we applied in our TREC Ad-Hoc, SDR and CLIR participations (Franz and Roukos, 1998), (Franz et al., 1999).

Relevance scoring was based on morph unigram and bigram features, extracted from the text data using a decision tree based tokenizer, part-of-speech tagger (Merialdo, 1990) and a morphological analyzer.

In the first pass, we used a modification of the Okapi formula (Robertson et al., 1995), described in (Franz et al., 1999) to score passages extracted from the encyclopedia documents. We converted the encyclopedia articles into 82277 overlapping passages, each containing approximately 100 non-stop words. Based on the first pass passage ranking, we constructed expanded queries using the local context analysis (LCA) technique (Xu and Croft, 1996), modified as described in (Franz et al., 1999). In the second pass scoring, we used the expanded queries to score 2632807 passages based on the TREC-9 Q&A corpus. The passages were selected to contained approximately 200 non-stop words.

Table 2 summarizes the information retrieval results on the 146 development test set questions described below. The performance is measured by the

Unigrams	What year did World War II start?
Morphed,Part-Of-Speech	what{WP} year{NN} do{VBD} World{NP} War{NP} II{NP} start{NN}
Bigrams	what{wp} what{wp}_year{nn} what{wp}_do{vbd} what{wp}_world{np} ...
Expanded Hierarchy	what{WP} year time_period measure abstraction year{NN} do{VBD} ...
Mark Question Word	what_bqw year time_period measure abstraction year{NN} do{VBD} ...

Table 1: Features used in the answer classification experiments

	MRR
pass1, TREC	0.4605
pass2, TREC	0.4824
pass2, encyclopedia	0.5031

Table 2: Retrieval results.

Mean Reciprocal Rank (MRR) (Voorhees and Tice, 1999) of the highest ranking passage containing the answer string among the top five passages. The first line of the table shows the result of first pass scoring using the TREC-9 Q&A database. The second line contains the result obtained with queries expanded using the TREC database. The last line of the table shows the result corresponding to the system applied in our official submission, with queries expanded using the encyclopedia database.

4 Named Entity Annotation

Named entity annotation is a markup of the text with the class information. As mentioned above, our classes correspond to the MUC classes due to the availability of training data for these classes. We used the text corpora available from the LDC to train the maximum entropy model.

Windows of +/- 2 words, morphs, part-of-speech tags and flags raised by pattern grammars for DATE, MONEY, CARD, MEASURE, PERCENT, TIME, DURATION classes, along with the two previous tags are created for each word. The window for predicting the tag(0) is shown in Table 3. Each stream has a fixed vocabulary and n-grams from this vocabulary form the features of the maximum entropy model. The training data is arranged to indicate a special category for beginning each named entity, for example BeginPERSON to find the boundaries of the named entity.

The system explores multiple NE hypotheses in parallel and keeps only those with high probability and proceeds with a beam-search algorithm to find the most likely path for the whole sentence. The performance of the named entity detector is comparable to the performance cited in (Borthwick et al., 1998) when training the maximum entropy algorithm on only annotated data. We omit the results here in the consideration of space, but note that in the analysis of the question answering system below only 4 out of 64 errors are attributed directly to the named entity marking for the 250 byte system.

Words	w(-) w(-) | w(0) | w(+) w(+)
Morphs	m(-) m(-) | m(0) | m(+) m(+)
Part-of-Speech	p(-2) p(-1) | p(0) | p(1) p(2)
Grammar Flags	f(-2) f(-1) | f(0) | f(1) f(2)
Previous Tags	t(-2).t(-1) t(-1) |

Table 3: Features used in the named entity model for predicting tag(0).

5 Answer Selection

We receive in this module the question, the class of the answer that the question seeks and a ranked set of passages (70) annotated with the MUC classes. The optimal sentence that answers the question is now sought. The TREC length constraints of 250 byte and 50 byte are then applied on the sentence.

The algorithm used in this module is listed here:

1. Each retrieved passage is split into sentences.
2. A window is formed around each sentence (window size is 3 sentences)
3. The following distances are computed: Matching Words, Thesaurus Match, Mis-Match Words, Dispersion, and Cluster Words. These are defined below.
4. The location or absence of the desired entities is noted in the score.

5. Each of these distances are weighted, the sentences ranked and the top 5 sentence are then output.

The definition of the various distances are

Matching Words The TFIDF sum of the number of words that matched identically in the morphed space. (+)

Thesaurus Match The TFIDF sum of the number of words that matched using a thesaurus match using WordNet synonyms ((Miller, 1990)). (+)

Mis-Match Words The TFIDF sum of the number of question content words that did not match in this answer. (-)

Dispersion The number of words in the candidate sentence that occur between matching question words. (-)

Cluster Words The number of words in the candidate sentence that occurred adjacently in both the question and answer candidate. (+)

Each distance has a weight applied and the corresponding sign shown above attached to it. The score for an answer is the sum of the distances and the top 5 sentences are then output.

To select the 250 or 50 byte answer chunk from these sentences, the system identified the longest mismatched pieces between the answer and the question. It then analyzed the answer and the question to find where the center of the match was and using a subject-verb-object assumption of the sentence, it took the question as either desiring the subject or object whichever had the least matches with the question.

Answer selection as done above used mostly heuristic distance metrics to seek an answer. Future work by the authors will show how to treat these distance metrics as features and to develop a statistical model for answer selection for an open domain.

6 Development Set Analysis

We wanted to maintain the TREC-9 database as a test set, but in order to do some post-evaluation analysis, we chose a subset of the questions as a development set for next year. There were two classes of questions in this years evaluation: questions that had only one phrasing and questions that had more than one phrasing (rephrased). We wanted 20% of questions of each class in the development test. The exact list of questions we used for our TREC-9 development test set are shown in Table 4. The variant questions we chose are shown in italics, and we added every seventh question skipping the ones in the above class to yield the 146 questions. A set of answer patterns was developed for the set using the judgements file provided by NIST.

201	203	209	210	217	220	224	231	238	242
245	252	253	259	264	266	273	275	280	286
287	294	297	301	308	315	319	322	329	330
336	341	343	350	352	357	363	364	371	374
378	385	392	393	399	411	412	413	420	434
453	454	456	458	462	469	473	476	483	484
490	495	497	504	506	511	517	518	525	528
532	539	546	550	553	560	561	567	572	574
581	583	588	594	595	602	605	609	616	623
627	630	637	638	644	649	651	658	660	665
671	672	679	682	686	693	700	*711*	*712*	*713*
714	*715*	*716*	*717*	*718*	*719*	*720*	*721*	*722*	*723*
724	*725*	*726*	*727*	*728*	*729*	*730*	*731*	*732*	*733*
734	*805*	*806*	*807*	*828*	*829*	*830*	*831*	*832*	*833*
834	*839*	*840*	*841*	*842*	*843*				

Table 4: Question numbers chosen for the TREC-9 development set.

System	TREC9 results	DEV WB expansion	DEV TREC expansion
250 byte	0.457	0.437	0.417
50 byte	0.290	0.287	0.266

Table 5: MRR for TREC-9 and the chosen dev set

The MRR for the entire system for the 250 byte system and the 50 byte system is shown in Table 5.

Analysis of the components are shown in Table 6. An error is attributed to a component if it is the first component that caused the failure working left to right in our system architecture. Fixing this error though need not correct the final answer as it may invoke an error in a subsequent system. Answer selection is still seen to be the major cause of problems in our question answering system.

Component	Number of Errors (Error rate)	
	250 byte	50 byte
Answer Type	5 (3.4%)	7 (4.8%)
IR	19 (13%)	19 (13%)
NE	4 (2.7%)	5 (3.4%)
Answer Selection	36 (24.7%)	52 (35.6%)
System	64(43.8%)	83(56.8%)

Table 6: Component error rate for the TREC9 dev set for 250 byte system

Q&A rank	IR rank						Total
	1	2	3	4	5	5+	
1	29	9	5	3	2	5	53
2	10	2	1	0	0	0	13
3	2	2	1	0	1	0	6
4	1	1	0	1	1	2	6
5	2	1	0	0	1	0	4
5+	13	7	2	1	1	40	64
Total	57	22	9	5	6	47	146

Table 7: Rank transition matrix, IR ws Q&A, 250 bytes

Q&A rank	IR rank						Total
	1	2	3	4	5	5+	
1	20	5	2	1	0	3	31
2	5	2	1	0	0	1	9
3	6	2	1	1	1	0	11
4	3	1	0	0	1	1	6
5	2	1	1	0	1	1	6
5+	21	11	4	3	3	41	83
Total	57	22	9	5	6	47	146

Table 8: Rank transition matrix, IR ws Q&A, 50 bytes

Another viewpoint is to see the effect of the system on the IR ranking results. This is shown below in Figure 3. Finding the 250 bytes from a passage that is of typical length 2.4K bytes shows some degradation, but further finding the 50 byte answer has considerable degradation. In Tables 7 and 8 we show the transition matrix for the rank from IR to the Q&A system. Note that there are significant transitions between the IR rank and the Q&A rank, but that inspection of the final result in Figure 3 shows that overall system performance is similar to the performance of IR for the 250 byte system and degraded at 50 bytes. This we believe points to the possibility of making more improvements in answer selection by reranking the results.

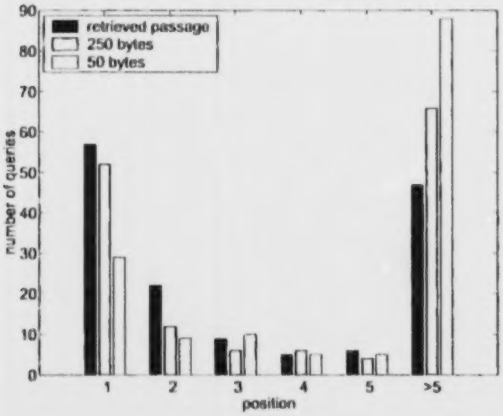


Figure 3: Development Set Performance

7 Conclusion

We presented above our architecture and a component wise evaluation of the system in the question answering problem. This was our first year of developing this system and having performed above the mean we believe that much more can be done in future evaluations. Our current work is to utilize maximum entropy features in the answer selection process which will render the system completely trainable from examples.

8 Acknowledgement

The authors thank Salim Roukos, Kishore Papineni and Todd Ward for making this work possible and helping us with their enormous expertise.

References

Adam Berger and Harry Printz. 1998. A comparison of criteria for maximum entropy/minimum divergence feature selection. *Proceedings of the Third Conference on Empirical Methods in Natural Language Processing*, pages 97–106, June.

Adam L. Berger, Vincent Della Pietra, and Stephen Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1).

A. Borthwick, J. Sterling, E. Agichtein, and R. Grishman. 1998. Exploiting diverse knowledge sources via maximum entropy in named entity recognition. *Proceedings of the COLING-ACL 98, Sixth Workshop on Very Large Corpora.*

Nancy Chinchor. 1997. Muc-7 named entity task definition. *Proceedings of MUC-7.*

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. 1995. Inducing features of random fields. *Technical Report CMU-CS-95-144*, May.

M. Franz and S. Roukos. 1998. Trec-6 ad-hoc retrieval. In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Sixth Text REtrieval Conference (TREC-6).* NIST Special Publication 500-240.

M. Franz, J. S. McCarley, and S. Roukos. 1999. Ad-hoc and multilingual information retrieval at ibm. In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*. NIST Special Publication 500-242.

B.F. Green, A.K. Wolf, C. Chomsky, and L.K. Baseball. 1963. An automatic question answerer. *Computers and Thought*, pages 207–216.

Academic Hallmarks. 1999. Knowledge master. *http://www.greatauk.com*.

B. Merialdo. 1990. Tagging text with a probabilistic model. In *Proceedings of the IBM Natural Language ITL*, pages 161–172.

G. Miller. 1990. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4).

S.E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. 1995. Okapi at trec-3. In D.K. Harman, editor, *Proceedings of the Third Text REtrieval Conference (TREC-3)*. NIST Special Publication 500-225.

Ellen M. Voorhees and Dawn M. Tice. 1999. The trec-8 question answering track evaluation. *TREC-8 Proceedings*, pages 41–63.

Jinxi Xu and W. Bruce Croft. 1996. Query expansion using local and global document analysis. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

IIT TREC-9 - Entity Based Feedback with Fusion

A. Chowdhury, S. Beitzel, E. Jensen, M. Sai-lee, D. Grossman, O.Frieder
Information Retrieval Laboratory
Department of Computer Science
Illinois Institute of Technology
Chicago, IL 60616
{abdur, beitzel, jensen, lee, grossman, frieder} @ ir.iit.edu

M.C. McCabe
Office of Advanced Analytical Tools
U.S. Government

D. Holmes
NCR Corporation
Rockville, Maryland
David.Holmes@WashingtonDC.NCR.COM

Abstract

For TREC-9, we focused on effectiveness in the web track. The key techniques we employed were information fusion, entity-based relevance feedback, Wordnet-based query parsing and a user interface designed to assist with web-based manual queries. Our initial results are positive. For the manual task, forty of fifty queries are over the median. In the adhoc, title-only task, thirty-four of fifty queries are over the median.

1. Introduction

For TREC-9 we focused on the web track, especially on in improving out effectiveness on large volumes of data. The past few TREC's we have focussed on scalability by treating the IR problem as an application of a parallel, relational database [Grossman97]. To focus on effectiveness this year, we built a new Java-based IR system called AIRE (Advanced Information Retrieval Engine). AIRE is designed to be a flexible, modular IR engine capable of state-of-the art retrieval techniques and easily modified to incorporate new proven or experimental techniques.

The keys to our effectiveness included the following approaches:

- Fusion using probabilistic (both traditional and models involving self-relevance [Robertson98, Kwok96], and vector space model with pivoted document length normalization [Singhal96].

- Entity-based relevance feedback [McCabe00].

- Improved parsing techniques of both the query and the documents.

- New user interface for manual queries.

Section 2 describes each of these approaches. Our initial results are positive. For the manual task, forty of fifty queries are over the median. In the adhoc, title-only task, thirty-four queries of fifty are over the median. More details on these results are given in Section 3. Section 4 describes directions for future work.

2. Approach to TREC-9

Our basic approach was to focus solely on the manual and adhoc parts of the web track. Given that our previous work focused on scalability, this year was a significant challenge as we started to really focus on effectiveness. We calibrated our techniques using the TREC-8 adhoc and web document collections. Early in the TREC-9 year, our best average precision was around 0.23 (this was about the average for effectiveness at TREC-8) and after applying a variety of fusion techniques we improved to roughly 0.28. At this point, we incorporated collection enrichment and then re-calibrated our fusion techniques. This left us at around 0.30. Finally, we improved our parsing and stemming algorithms using a combination of our own modified Porter stemmer and the U-Mass conflation classes [Xu96, Xu98, Pickens]. At this point we were at around .31 for the adhoc (disks 4 and 5) and .36 for the TREC-8 small web track. At about this time, the TREC-9 queries came out and we ran our best calibrated system against the TREC-9 collection. Details of our fusion techniques are given in Section 2.1, details of our parsing techniques are given in Section 2.2.

Additionally, we experimented with a means by which we could integrate information extraction with information retrieval. The idea of this technique is to use entities identified by an information extractor (we used SRA's) system and then *only* add entities in the feedback process. We submitted one adhoc run without the use of entities: *iit00td* (title + description), and *iit00t* (title) and one run with the use of entities *iit00tde* (title + description + entities). Such feedback with extraction is described in Section 2.3.

For the manual track we built a new user interface to facilitate manual query processing. Details of this user interface are given in Section 2.4.

Fusion

Prior work in fusion combined results from disparate retrieval systems [Fox94, Bartell94, Lee97]. Our approach was to provide fusion via one common system. Using a common parser, stoplist, inverted index, etc, we implemented a variety of retrieval algorithms within our framework. Thus, we avoid confusing fusion improvements with simple parsing or other system differences. We conducted numerous calibrations using the vector space model [Singhal96], Robertson's probabilistic retrieval strategy [Robertson98], and a modified vector space retrieval strategy. The following equations describe those used as the foundation of our retrieval strategies.

Robertson's Retrieval Status Value (RSV)

$$RSV = \sum_{T \in Q} w \left(\frac{(k_1+1)tf}{(K+tf)} \frac{(k_3+1)qtf}{k_3+qtf} + k_2 |Q| \frac{avdl-dl}{avdl+dl} \right)$$

where tf = frequency of occurrences of the term in the document
qtf = frequency of occurrences of the term in the query
dl = document length
$avdl$ = average document length
k_n are parameters set based on the nature of the queries and the collection.

$$\sum \log\left(\frac{N-n+.5}{n+.5}\right)\left(\frac{(2.2)tf}{.3+(.75 * dl/avdl)+tf}\right)qtf$$

Our implementation with constants as specified by Robertson

Singhal's Similarity Coefficient

$$SC(Q,D_i) = \frac{\sum_{j=1}^{t} q_{qj} d_{ij}}{((1.0-s)p + s(|d_i|))}$$

where $|d_i|$ is the number of elements in the vector, or the number of distinct terms in the document, s is the *slope* which Singhal calculated for a variety of test corpuses (mostly TREC subsets) and found that 0.20 works well across most collections. The pivot, *p*, is the point, or document length at which the probability of relevance equals the probability of retrieval. This is estimated to be the average document length of the collection.

In addition to fusion of various retrieval strategies, AIRE permits the fusion of different *query representations*. Also, each input run has a scalar weight that indicates the relative importance of the run.

For our first pass retrieval, we focused on finding one retrieval strategy that did better for high recall and another strategy that performed well for high precision (at 30 documents). Our hypothesis was that the combination would perform better in terms of average precision than either input run. Our initial results showed a slightly modified Vector Space did well for high recall and that Robertson's probabilistic model did the best at for precision at 30. The combination we ultimately settled on was: Modified Vector Space title-only (weighted at 1.0), Robertson title-only (weighted at 0.1), Robertson description when applicable (weighted at 0.7). This fusion combination effectively emphasized title terms as most important while still benefiting from high-recall description terms.

For our second pass, we selected the top fifteen feedback terms from the top ten documents using the fused pass one run. In order to select the top fifteen terms, we first weighted each term found in the top documents using Robertson's term weighting. We then calculated the ultimate rank of the candidate term using Rocchio's relevance feedback formula [Rocchio71]. In addition to finding the top fifteen terms and phrases, a check is made to a list of nouns obtained from Wordnet to filter candidate terms and phrases so that only nouns are selected. The new query terms are then used in pass two as one of the query representations for a fusion input run. We found a scalar weight of .5 and the Roberston retrieval strategy to work well with this query representation.

We also used a collection enrichment representation for a pass two fusion input run. This query run consisted of terms selected from a pass one retrieval executed against the TREC disks 4 and 5. The fifteen top ranked terms are then used as a query against the search collection (10GB web) with Robertson retrieval strategy and a weight of .5 (same as relevance feedback terms).

Finally, two additional runs are included in the pass two fusion. The original title terms are used with modified vector space retrieval weighted at 1.0. The original description terms are used with Robertson weighted at 1.0.

Interestingly enough, for our final TREC submission, we did not normalize the fusion runs. Thus our scalar weights represent actual multipliers rather than relative importance in pass two. This choice was made based on prior calibrations.

In summary, our *iit00t*, *iit00td* and *iit00tde* submissions were fusions of the following four different representations of a single query:

- Title words only
- Description words only (this was only used for runs involving the description)
- Relevance feedback terms obtained from running the title (and description when applicable)
- Relevance feedback terms (and entities for tde) obtained from a collection enrichment run (TREC disks 4 and 5)

Information Extraction

In previous years, our manual runs did well when the user added person and place names to queries. For example *Kuhn Sa* was very helpful on the query regarding drug triangle. This year, we propose entity-based feedback as a method to automatically select such person names, as well as place names and organization names and add them to the query. The technique required modifications to the inverted index in order to include term-type (term, phrase, person, location, etc.). Secondly, the document preprocessing was modified to include SRA's Name Tagger to identify entities within the text. Then, the original query of only terms and phrases was run for pass one. Pass two selects entities from the top documents returned and adds these terms to the query as in relevance feedback.

For our calibrations, we isolated the entities and added only person names, only locations, and only organizations to the query. In order to understand the real impact of each entity, we ran many calibrations where only one new word was added to the query. We found that good improvement is possible when we adding only a single organization, a single location, or a single person name. Details of these calibrations may be found in [McCabe00]. For example, query *Ireland Peace Talks* added the organization *Sinn Fein*, the person *Jerry Adams*, and the location *Northern Ireland*. Each of these improved the query effectiveness by over 100%. While more queries improved than degraded with each entity type, several queries degraded badly. Names that were associated with more than the query topic were harmful. For example, the query *Estonian Economics* selected the Estonian Prime Minister Mart *Laar* as the name to add. This degraded performance because the prime minister is in many documents having nothing to do with economics. In addition, ambiguous names were harmful. It turns out there are many individuals with the name *Stirling*. So that addition to the query *Stirling Engine* brought back documents about a California senator, a minister, etc.

For TREC-9 we selected entities from our collection enrichment corpus rather than our search corpus. That is simply because we already had our collection enrichment corpus (TREC disks 5 and 6) tagged and indexed with entities, while we had not yet tagged the web 10GB. In order to reduce the chances of a bad entity being selected, we added entities into the mix of potential feedback terms and only selected those that ranked in the top 15 terms or phrases. Our entity-based feedback run performed about the same as our title plus description run. This is because many queries did not select entities and of those that did some improved and some degraded, mostly canceling out the overall effect.

Parsing

We improved our parsing algorithms in TREC-9. Previously, we did not use any stemming. This year, we indexed terms with a modified Porter stemmer that does both prefix and suffix stemming. In addition, we use equivalence classes based on term co-occurrence to further restrict the stemming [Xu96,Pickens, Xu98]. If the term is found in the conflation file (a set of terms with their equivalence classes), we use the first occurrence of the term as the root form. If the term is not found, the modified porter stemmer is used. This technique corrects for over-stemming of common words. For example the standard Porter stemmer would conflate *policy* and *police* while these equivalence classes would not.

In addition to single terms, our parser indexes standard statistical two-term phrases. Like numerous groups over the years, a sliding two-term window is used to detect these phrases. Any span punctuation or stop term prevents a phrase. We also eliminate phrases that do not occur more than 25 times.

In order to update our parser to accommodate web-type queries such as misspelled and mis-spaced terms, we incorporated a "find a real query term" algorithm using Wordnet. Our algorithm finds the longest common sub-string match in the query that is also a noun in Wordnet. If the initial query found no documents, we use this algorithm as an automatic best guess approach.

Manual Query Processing

Our previous years at TREC have shown that a user who is given the ability to add related terms to a "concept" for a query is able to improve effectiveness. Our query expert now has six years of experience with TREC queries and is quite comfortable with defining terms for a query. About 5-10 minutes are spent on each query in which two different concepts are defined for "inclusion" into the query and one is defined for "exclusion". The ultimate query may be expressed such that if terms in the set C1 $\{c_{11}, c_{12}, \ldots, c_{1n}\}$ and the set C2 $\{c_{21}, c_{22}, \ldots, c_{2n})$ are included and terms in the set X $\{x_1, x_2, \ldots, x_n\}$ are excluded, the following Boolean processing is done on the query: ((c_{11} OR c_{12} OR …., OR c_{1n}) AND (c_{21} OR c_{22} OR ….OR c_{2n})) NOT (X_1 or X_2 or X_n). Additionally, for other related terms that are not used to filter a document a scoring concept is used. For these terms S $\{s_1, s_2, \ldots, s_n\}$, only the similarity measure is affected – these terms are not used to filter the document. Once the Boolean filters are incorporated, standard tf-idf VSM is employed to rank documents.

A Java servlet is used to provide quick feedback to the user. For each initial request, the user quickly views documents obtained in response to the request. Additionally, relevance feedback terms and phrases are suggested. Overall, our test user was quite pleased with the new user interface (only command line SQL processing has been available in previous years).

3. Results

We describe our adhoc results first, then our manual results, and finally we give some initial failure analysis.

Adhoc

Our title-only run was called *iit00t* and our title with description run was named *iit00td*. The run which used named entities as part of the collection enrichment was entitled *iit00tde*. The table below gives a summary of our results. The columns indicate the average precision for the median of all groups, IIT's average precision, the number of queries at or above the median, the number of queries below the median, the number of queries that gave the best results, and the number of queries that gave the worst results.

Run	**Avg. Median**	**IIT Avg. Precision**	**# Above Median**	**# At Median**	**# Below Median**	**# Best**	**# Worst**
iit00t	.1212	.1627	30	2	18	5	4
iit00td	.1554	.2227	38	1	11	3	0
iit00tde	.1554	.2293	37	2	11	4	1

Manual

For the manual query track, we had promising results. With all but seven queries over the median and twenty-five queries listed as achieving the highest average precision, we are pleased with this run. Unfortunately, one query was found to be the worst.

Run	**Avg. Median**	**IIT Avg. Precision**	**# Above Median**	**# At Median**	**# Below Median**	**# Best**	**# Worst**
iit00m	.1350	.3519	40	3	7	25	1

Failure Analysis

In failure analysis, we review poor performing queries and analyze the cause of failure. We started some failure analysis for TREC-9. The manual query with the worst average precision (0.000) was topic 485 which simply contained the terms "*gps clock*". For this query, there were only two relevant documents and we did not find either of them in our top 100. The reason is that we added numerous synonyms for gps and for clock and they overshadowed the basic phrase "gps clock." Worse, our adhoc system stemmed "gps clock" to "gp clock." Once we stemmed "gps" to "gp" we found documents about "general permit" (Document wtx082 -b37-24) and *grand* *prix*, hockey statistics for *games played*, etc. A simple rule that precluded stemming three character terms would have improved this run tremendously. It is not clear how we would know that our manual run could be improved, but one approach might be to simply run the manual query and fuse it with the entire original query.

For topic 495, *Where can I find information on the decade of the 1920's?*, our user tried to think of events in the 1920's that would be of interest (e.g.; Charles Lindbergh, Calvin Coolidge, etc.). Unfortunately, he missed many useful events in the 1920's and missed some relevant documents.

In addition, our analysis indicates that our use of a scoring concept in the manual runs hurt us for some queries. It may well be helpful to use fusion to combine a query run with the scoring concept and without the scoring concept so as to ensure that high scoring documents without the scoring concept are included in the final result set.

4. Summary and Future Work

Overall, we are pleased with our work on effectiveness this year. We plan to spend more time on failure analysis. Additionally, more cleanup of our parser is needed. More importantly, the potential of entity-based relevance feedback needs more research.

5. References

[Bartell94] Bartell, B. T., G.W. Cottrell, and R.K. Belew. Automatic combination of multiple ranked retrieval systems. *SIGIR '94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, 1994.

[Croft95] W. B. Croft and Jinxi Xu. Corpus-specific stemming using word form co-occurence. *In Proceedings for the Fourth Annual Symposium on Document Analysis and Information Retrieval*, pages 147--159, Las Vegas, Nevada, April 1995.

[Fox94] Fox, E. and J. Shaw. "Combination of Multiple Searches, *Proceedings of the 2nd Text Retrieval Conference (TREC2),*National Institute of Standards and Technology Special Publication 500-215, 1994.

[Gross97] Grossman, D., O. Frieder, D. Holmes and D. Roberts, Integrating Structured Data and Text: A Relational Approach, *Journal of the American Society for Information Science*, January 1997.

[Kwok96] Kwok, K.L (1996). A new method of weighting query terms for ad-hoc retrieval. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.187-195.

[McCabe2000] McCabe, M.C. *Improving Information Retrieval Effectiveness with Databases, Fusion and Entity-Based Feedback.* GMU PhD thesis. Aug.2000.

[Lee97] Lee, J.H. Analysis of multiple evidence combination. *SIGIR '97: Proceedings of the Twentieth Annual InternationalACM-SIGIR Conference on Research and Development in Information Retrieval*, 1997.

[Porter80] Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3):130--137.

[Robertson98] Robertson S., S. Walker and M. Beaulieu, Okapi at TREC-7: Automatic ad hoc, filtering, VLC and Iinteractive. *In Proceedings of the Seventh Text Retrieval Conference (TREC)*7, 1998.

[Singhal96] Singhal, A., C. Buckley, and M. Mitra, Pivoted Document Length Normalization. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1996.

[Xu98] Jinxi Xu and W. Bruce Croft. Corpus-based stemming using co-occurrence of word variants. Technical Report TR96-67, Dept. of Computer Science, University of Massachusetts/Amherst.

[Pickens] Jeremy Pickens, "Stemming and Cooccurrence on a Larger Corpus"

[Rocchio71] J. Rocchio. Relevance Feedback in Information Retrieval. *Smart System - Experiments in Automatic Document Processing*, pages 313--323. Prentice Hall, Englewood Cliffs, NJ, 1971.

[Xu96] J. Xu and W. Croft, Query Expansion Using Local and Global Document Analysis. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 4--11, 1996.

A Simple Question Answering System

Richard J Cooper and Stefan M Rüger
Department of Computing
Imperial College of Science, Technology and Medicine
180 Queen's Gate, London SW7 2BZ, England
s.rueger@doc.ic.ac.uk

Abstract. We describe our simple question answering system written in perl that uses the CMU Link parser (Sleator and Temperley 1991), Princeton University's WordNet (Miller 1995), the REX system for XML parsing (Cameron 1998) and the Managing Gigabyte search engine (Witten, Moffat and Bell 1999). This work is based on an MSc project (Cooper 2000).

Introduction

The main task of question answering is providing a short answer to a natural-language query supported by a document in an underlying document collection. Many question-answering systems approach the problem from an information extraction angle, and ours is no exception. In the following, we will describe the structure and workings of our system. Section 1 is concerned with the off-line preparation of the documents for the pure information retrieval task of identifying potentially relevant paragraphs. Section 2 details the question-time processing.

1 Indexing

All documents from the document repository[1] are cleaned from SGML mark-up to obtain their raw contents. Dollar and pound signs are replaced by the words *dollars* and *pounds*, respectively (in line with the Financial Times archive which already uses these transformations). The raw news articles are then split into paragraphs according to the individual newspaper's indication. This could be an indentation after a newline (AP newswire, Foreign Broadcast Information Service and Wall Street Journal), a special marker (San Jose News), the use of a long line per paragraph (Los Angeles Times), or the fact that a line is not right justified that signifies the end of a paragraph (Financial Times).

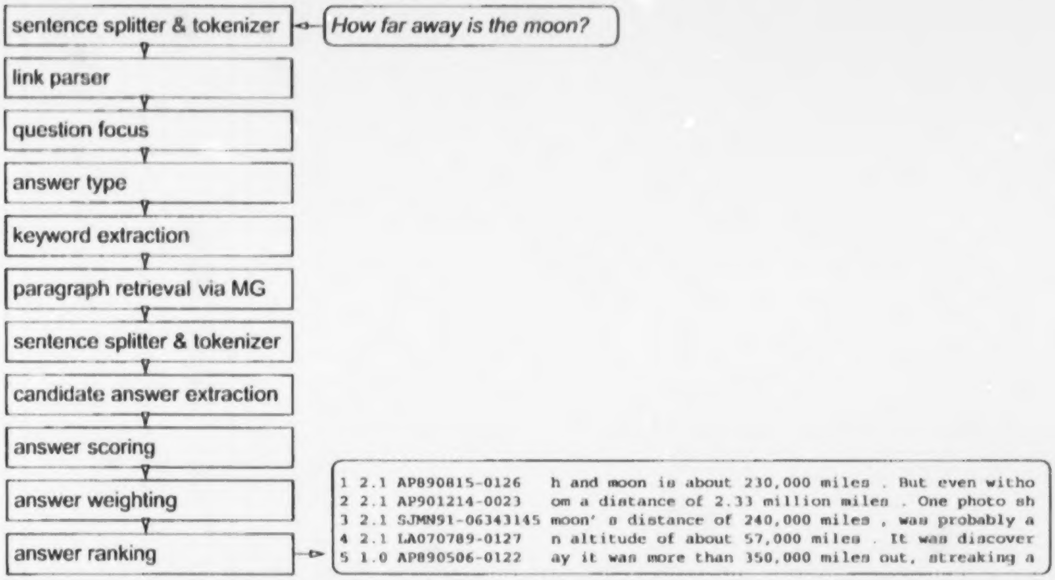
The paragraphs are then fed into the Managing Gigabytes search engine as stand-alone documents together with the original reference number. In effect, we get a passage retrieval rather than a document retrieval.

2 Question processing

The actual question processing is executed as a long pipeline of perl modules which use XML, eg, to mark-up entities or to communicate other information between the modules. At the start of the pipeline is the question, eg, *Q206: How far away is the moon?* Figure 1 shows the structure of the data flow and the modules involved.

---

[1] News articles from TIPSTER and TREC CDs, see http://trec.nist.gov


sentence splitter & tokenizer
How far away is the moon?
link parser
question focus
answer type
keyword extraction
paragraph retrieval via MG
sentence splitter & tokenizer
candidate answer extraction
answer scoring
answer weighting
answer ranking

```
1 2.1 AP890815-0126   h and moon is about 230,000 miles . But even witho
2 2.1 AP901214-0023   om a distance of 2.33 million miles . One photo sh
3 2.1 SJMN91-06343145 moon' s distance of 240,000 miles , was probably a
4 2.1 LA070789-0127   n altitude of about 57,000 miles . It was discover
5 1.0 AP890506-0122   ay it was more than 350,000 miles out, streaking a
```

Figure 1: Question processing: data flow

2.1 Sentence splitter & tokenizer

The sentence splitter and tokenizer are actually implemented as two modules. The first marks up individual sentences using a set of heuristics for detecting the end of a sentence by a question mark, an exclamation mark, a full stop (if not preceded by a word in a list of abbreviations, such as Mr and Mrs), or the end of text. The tokenizer treats certain leading and trailing punctuation as separate entities, eg, "(", "[", ")", "]" or "?". Words containing digits are separated, if headed by a currency such as *DM* or *pounds* or trailed by an abbreviation such as *m*. Hence, *pounds20m* becomes the tree tokens *pounds*, *20* and *m*.

2.2 Link parser

The link parser is used to annotate the structure of the question. The link tree is appended to the tokenized question. For Q 206, eg, we get the following

```
<sentence><t n="1">How</t> <t n="2">far</t> <t n="3">away</t> <t n="4">is</t>
<t n="5">the</t> <t n="6">moon</t><t n="7">?</t><parse><pos n="2" pos="a"/><pos
n="4" pos="v"/><pos n="6" pos="n"/><link name="Xp" l="0" r="7"/><link name="Wq"
l="0" r="2"/><link name="PF" l="2" r="4"/><link name="MVp" l="2" r="3"/><link
name="SIs" l="4" r="6"/><link name="Ds" l="5" r="6"/><link name="RW" l="7"
r="8"/></parse></sentence>
```

BEST COPY AVAILABLE

2.3 Question focus

We identify the following question types which give a fairly clear indication of the type of answer: *when, who, where, whom, why, describe,* and *define*. For example, the answer to a *who* question is usually a person or a group of people. Other keywords or question words are less clear about the expected answer type: *what, which, how,* and *name*. For example, consider the following three *what* questions: *What **time** is the train arriving?*, *What **city** is the train stopping at?*, and *What is the name of the **driver** of the train?* This problem can be solved by defining a concept called question focus. The question focus is a phrase in the question that disambiguates it and emphasizes the type of answer being expected. For example in the three questions above the question foci are shown in bold. In the first two cases, the question focus tells us directly that a time and a city are being looked for. In the third case we know that a driver is a type of person and hence that a person's name is being sought. For the purpose of this system the question focus is defined as the first noun group that is not the word "name" if the question word is of an ambiguous type.

Normally question words start a question such as *when* in *When was Queen Elizabeth II born?* However sometimes they do not, eg, *In what city is the US Declaration of Independence located?* and *Macintosh Computers are made by whom?*, and we consider these cases as well.

For question 206, the mark-up `<questionFocus><t n="6">moon</t></questionFocus>` is inserted.

2.4 Answer type

Once the focus of the question has been found it is possible to decide what the answer type (or answer concept) should be. The question's question word determines how this module uses the question focus.

2.4.1 When, where, why, describe, define

When, where, why, describe and *define* are the easiest question words to process. They map directly into their answer type. The answer types for these question words are time, place, reason, description and definition, respectively.

2.4.2 Who, whom

In most cases *who* and *whom* question words imply an answer concept of person. However, there are two subtleties that are worth mentioning at this point.

Firstly, who questions could be looking for a group of people. This group could be a named group: *Who won the Premiership?* (Manchester Utd) or a list: *Who beat Fred in the 100m?* (Tom, Dick and Harry) or a combination: *Who beat England in the relay?* (America and Canada). At this stage the system does not correctly handle such questions, instead it always looks for a single person in response to a who question.

Secondly, there is an important exception to the rule that who questions always have an answer concept person. Consider, for example, *Who is Bill Gates?* This is an example of a who question that is looking for a description rather than a person. Accordingly, the rules for who and whom have the exception `(who|whom) (is|are|was) ProperNoun` which returns description as the answer type.

2.4.3 What, which, name

What, which, and *name* question words are extremely ambiguous when it comes to determining the question's answer concept. The answer type for these questions depends entirely on the question focus. The following pseudo-code illustrates the algorithm used:

```
until(wordNet contains questionFocus or questionFocus = "")
    Remove the first word of questionFocus
end until
if(questionFocus = "") then
    answerConcept = "name"
    exit
end if
hyponyms = the set of hyponyms of questionFocus from wordNet
if("person" is in hyponyms) then
    answerConcept = "person"
else
    answerConcept = questionFocus
end if
```

Following this idea further, one could also define the answer type as the list of direct hyponyms. Given the question *What type of bridge is The Golden Gate Bridge?*, the answer type *bridge* could be expanded (using WordNet) to include Bailey bridge, cantilever bridge, covered bridge, drawbridge, lift bridge, footbridge, overcrossing, pedestrian bridge, gangplank, gangboard, gangway, overpass, flyover, flypast, pontoon bridge, bateau bridge, floating bridge, rope bridge, steel arch bridge, suspension bridge, trestle bridge, truss bridge, or viaduct.

In our example of Q 206, `<answerType t="Length"/>` is added to the data stream.

2.4.4 How

With *how* questions the answer type is defined by what the word after *how* is — for example: *how old* (age), *how much* (quantity), *how long* (distance). If the word after *how* does not match any of the rules then the default answer type of *manner* is chosen, as in: *How did Socrates die?*

2.5 Keyword extraction

Using lists of place names, proper names and first names, entities are recognized and marked up with special symbols. This is simple non-structured knowledge that goes as far as identifying UK as a country, London as a city and Tony Blair as a person but the lists do not encode the relationship between these entities (and hence it is not known a priori that London is the capital of the UK). Recognized entities are among speed, temperature, money, place, city, country, person, year, time, length, reason, company, number, quoted and name, with the obvious meaning for the marked-up entity. In order to avoid inferences with potentially predefined SGML tags from the sources, we use special mark-ups of the form `<rjc99Person> Tony Blair </rjc99Person>`.

2.6 Paragraph retrieval via Managing Gigabytes

At the document paragraph lookup stage the information of answer type and possible keywords is used to extract documents (ie paragraphs) from the text database that might contain answers to the question.

The problem that the lookup stage has to overcome is the balance between getting sufficient documents to guarantee the presence of the answer and getting too many to process in a timely fashion. This problem defines the main task of the document paragraph lookup stage, which is to choose a "good" subset of the possible keywords with which to actually query the database.

This system defines "good" subsets by how many documents they retrieve. It attempts to find a subset that returns a number of documents within a given bound. As soon as such a subset is found it is chosen as the final query and the rest of the search is abandoned. If, after a complete search, no such subset is found, then the subset that is closest to that range is chosen. It attempts to reduce the amount of searching that need to be done by choosing the most likely subsets first, as defined by the weights assigned to each keyword. Once it has found a good subset, this module will query the document paragraph repository for real and add the texts retrieved to the output stream.

2.7 Candidate answer extraction

The paragraphs extracted from the document repository are sentence-split and tokenized in preparation for further processing. The Candidate Answer Extraction module's job is the mark-up of any regions of the texts that could be answers. The question's answer concept is looked up in WordNet and all of its hyponyms are found. A regular expression is then built by taking the disjunction of those hyponyms and any region of text that matches that regular expression is marked up as a candidate answer. There are a few exceptions to this rule which are detailed in the following sections.

2.7.1 Person

If the answer concept is person, the set of hyponyms would includes terms such as consumer, contestant, coward, creator, defender, guardian, and over 300 other words. This is not the intent as a person question is looking for a specific name of a person. So in the case of person questions, the hyponyms set is not computed and a regular expression that matches names is used instead.

2.7.2 Description

Describe, define and some who, whom and what questions result in an answer concept of description. Unlike answer types such and person and date, descriptions are very hard to define in terms of what words make them up. Any attempt to mark-up the descriptions in a piece of text would have to employ much more sophisticated NLP techniques than used in this system. Consequently, this system uses a much simpler technique to extract regions of text that could be descriptions. It has been noticed that when an entity is first introduced in a text it is often followed by a comma and then a description, as in *Bill Gates, Head of Microsoft said today* ... In light of this, descriptions are defined as everything between a comma and the next punctuation mark. Then when it comes to scoring the answers, descriptions that are immediately preceded by the thing that they are describing are scored highly. Even using such a simple and naïve approach, good results are possible. For example, the top answers for the three questions below are shown in brackets: *Who is Steve Jobs?* (Co-founder and former chairman of Apple Computer), *Who is Steve Redgrave?* (Olympic gold medallist in 1984 and 1988), *Who is Nelson Mandela?* (President of the African National Congress).

2.7.3 General Cases

WordNet's coverage could never be great enough to cover every possible answer, especially with answer types that could contain an infinite number of answers, such as length. To deal with these eventualities, many of the common answer types have had extra subtypes added to WordNet which describe a regular expression for a general case. So, for example, *company* will match any of the explicitly named companies or the general case of anything that is a sequence of Proper Nouns ending in `(Ltd|Plc|Co|and Son|...)` and length will match any number followed by a unit of length such as *miles*, *km*, *ft*, etc.

2.8 Answer scoring

Once the candidate answers have been identified, a variety of heuristics are used to evaluate how likely that candidate is to be the real answer.

The following heuristics are used: (i) *score_comma_3_word*. If a comma follows the candidate answer then this score is the number of the three words following the comma that appear in the question; (ii) *score_punctuation*. Scores one if a punctuation mark immediately follows the candidate and zero otherwise; (iii) *score_same_sentence*. Computes the number of question words that are in the same sentence as the candidate answer; (iv) *score_description_before*. If the answer concept being looked for is a description then this score is the number of words immediately preceding the candidate answer that appear in the question; (v) *score_description_in*. Similar to *score_question_before* but counting question words that appear in the candidate answer.

Each heuristic is given a unique identifier, and at the end of this process each candidate will have associated with it a set of (id, score) pairs. Like most other things in this system, the scoring heuristics are implemented as pipeline modules. Each heuristic is a different module which scans its input for `<ca>` tags (for candidate answer), computes the appropriate score and adds a `<score>` tag. Each scoring heuristic is completely independent from the rest because, at this stage of the processing, the scores are being kept separate without regard for a single final score. This means that the order in which they activate is unimportant and that they can be removed or new ones added without affecting the others.

2.9 Answer weighting

Once all of the scores have been calculated they need to be combined into one final score. In this system the final score is simply a linear combination of all of the heuristic scores where the coefficients have been set by hand to reflect the perceived importance of the various scores: weight(*score_comma_3_word*) = 1.2, weight(*score_punctuation*) = 1.1, weight(*score_same_sentence*) = 1.0, weight(*score_description_before*) = 2.0, weight(*score_description_in*) = 1.0.

2.10 Answer ranking

Once every candidate answer has a weight they (and some of the characters around them which are to be used as context in the final answers) are extracted from the text, sorted by weight, assigned a rank and placed as final answers outside the document text.

At this stage one more thing happens. If the set of answers contains any duplicates then only the top ranking one is kept and all the other duplicates are removed. There are two possible modifications that could be made to this process that were considered, but not included in this version of the system. Firstly, only answers that are identical to higher ranked ones are removed. Ideally this would be replaced by a more powerful system which removed any answers that referred to the same entity as the first answer. A strong version of this system would probably require anaphora resolution of the document collection before they were split into paragraphs, and hence is beyond the scope of this system.

A weaker version, which removed any answers that probably referred to the same entity as the first answer, could be implemented using simple rules of abbreviation and word substitution. For example, Ms. L. Smith could be the same person as Miss Linda Smith, who could also be just Linda.

Alternatively, as some of the TREC-8 systems suggested, the presence of multiple instances of the same answer could be used to strengthen the likelihood of that answer being correct. We did not implement this.

3 Conclusions

Despite the apparent simplicity of our system, it compared favorably against other systems competing in TREC-9. Little use was made of natural language processing. The link parser analysis was only used for the proportion of questions that deal with ambiguous answer types; it was not used for the candidate answer extraction. We did not tune our system much, given the few training cases from the TREC-8 QA track. Although we did not (yet) carry out a failure analysis with the TREC-9 questions, we have reason to assume that there is scope for improvement in changing parameters, introducing a better ranking mechanism, or in deploying natural-language processing techniques.

References

Cameron, R. D. (1998). REX: XML shallow parsing with regular expressions. Technical Report 1998-17, School of Computing, Simon Fraser University.

Cooper, R. J. (2000). *High-Precision Information Retrieval.* MSc Thesis, Imperial College.

Miller, G. (1995). WordNet: A lexical database for English. *Communications of the ACM 38*(11), 39–41.

Sleator, D. and D. Temperley (1991). Parsing English with a link grammar. Technical Report CMU-CS-91-196, Computer Science, Canegie Mellon University.

Witten, I. H., A. Moffat and T. C. Bell (1999). *Managing Gigabytes.* Morgan Kaufmann Publishers.

Acknowledgements: This work was partially supported by the EPSRC, UK.

Mercure at trec9: Web and Filtering tasks

M. Abchiche, M. Boughanem, T. Dkaki, J. Mothe, C. Soule Dupuy, M. Tmar

IRIT-SIG
Campus Univ. Toulouse III
118, Route de Narbonne
F-31062 Toulouse Cedex 4
Email: trec@irit.fr

1 Summary

The tests performed for TREC9 focus on the Web and Filtering (batch and routing) tracks. The submitted runs are based on the Mercure system. For one of the filtering routing runs, we combine Mercure with mining text functionnalities from our system Tétralogie. We also performed some experiments taking hyperlinks into account to evaluate their influence on the retrieval effectiveness, but no runs were sent.

Web: We submit four runs in this track. Two elements were tested: a modified Mercure term weighting scheme and the notion of the user preference on the retrieved document were tested.

Filtering (batch and routing): our main investigation this year concerns the notion of non-relevant profile in a filtering system. The filtering consists on, first filtering the documents using a relevant profile learned from relevant documents, second re-filtering the selected documents using non-relevant profile learned from non-relevant documents so that non-relevant documents accepted by the relevant profile are rejected. This notion of non-relevant profile was introduced by Hoashi [6] in an adaptive system whereas we use this technique for a batch system.

2 Mercure model

Mercure is an information retrieval system based on a connectionist approach and modeled by a multi-layered network. The network is composed of a query layer (set of query terms), a term layer (representing the indexing terms) and a document layer [4],[3].

Mercure includes the implementation of a retrieval process based on spreading activation forward and backward through the weighted links. Queries and documents can be used either as inputs or outputs. The links between two layers are symmetric and their weights are based on the $tf-idf$ measure inspired from the OKAPI and SMART term weightings.

- the query-term (at stage s) links are weighted as follows:

$$q_{ui}^{(s)} = \begin{cases} \frac{nq_u * qtf_{ui}}{nq_u - qtf_{ui}} \; if \; (nq_u > qtf_{ui}) \\ qtf_{ui} \; otherwise \end{cases} \tag{1}$$

Notations:
$q_{ui}^{(s)}$: the weight of the term t_i in the query u at the stage s,
qtf_{ui}: the query term frequency of t_i in q_u,
nq_u: query length (number of terms) of q_u,

- the term-document link weights are expressed by:

$$d_{ij} = \frac{tf_{ij} * (h_1 + h_2 * log(\frac{N}{n_i}))}{h_3 + h_4 * \frac{dl_j}{\Delta d} + h_5 * tf_{ij}} \tag{2}$$

Notations:
d_{ij}: term-document weight of term t_i and document d_j
tf_{ij}: the term frequency of t_i in the document D_j,
N: the total number of documents,
n_i: the number of documents containing term t_i,
h_1, h_2, h_3 and h_4: constant parameters,
Δ_d: average document length.

The query evaluation is based on spreading activation. Each node computes an input and spreads an output signal. The query modification is based on relevance backpropagation. It consists in spreading backward the document relevance from the document layer to the query layer [3].

2.1 Query evaluation

A query is evaluated using the spreading activation process described as follows:

1. The query u is the input of the network. Each node from the term layer computes an input value from this initial query: $In(t_i) = q_{ui}^{(}s)$ and then an activation value: $Out(t_i) = g(In(t_i))$ where g is the identity function.
2. These signals are propagated forwards through the network from the term layer to the document layer. Each document node computes an input: $In(d_j) = \sum_{i=1}^{T} Out(t_i) * d_{ij}$ and then an activation , $Out(d_j) = g(In(d_j))$.

   The set of retrieved documents, $Output_u(Out(d_1), Out(d_2), ..., Out(d_N))$ is then ranked in a decreasing order of the activation value.

2.1.1 Query modification based on relevance backpropagation

The top retrieved documents are judged and a *relevance value* corresponding to the user preference is assigned to each document (positive for relevant documents, negative for non-relevant documents and nil for non-judged documents). These values are used to compute the *DesiredOutput* vector.
$DesiredOutput = (rel_1, rel_2, ..., rel_N)$, $rel_j = \frac{Coef\_rel}{Nb\_rel} for\ relevant\ document$ and $rel_j = \frac{Coef\_rel}{Nb\_Nrel} for\ non - relevant\ document$

1. This output is in fact considered as an "input" in the back-spreading process and is presented to the document layer. Each document node computes an input value, $In(d_j) = rel_j$ and then an activation signal, $Out(d_j) = g(In(d_j))$.

2. This activation is back-spread to the term layer. Each term node computes an input value, $In(t_i) = \sum_{j=1}^{N}(d_{ji} * Out(d_j))$ and then an output signal, $Out(t_i) = g(In(t_i))$.

3. Finally, the new query-term links corresponding to the new query are computed as follows: $Q_u^{(s+1)} = (q_{u1}^{(s+1)}, q_{u2}^{(s+1)}, ..., q_{uT}^{(s+1)})$ with $q_{ui}^{(s+1)} = M_a * q_{ui}^{(s)} + M_b * Out(t_i)$

   Notations:
   T: the total number of indexing terms,
   N: the total number of documents,
   $q_{ui}^{(s)}$: the weight of the term t_i in the query u at the stage s,
   t_i: the term t_i,
   d_j: the document d_j,
   d_{ij}: the weight of the link between the term t_i and the document d_j,
   $doclen_j$: document length in words (without stop words),
   avg_doclen: average document length, tf_{ij}: the term frequency of t_i in the document D_j,
   n_i: the number of documents containing term t_i,
   qtf: query term frequency,
   nq: query length (number of terms),

   M_a and M_b: tuned and determined by a series of experiments and set to $M_a = 2$ and $M_b = 0.75$.

   $Coef\_rel$ ($Coef\_Nrel$): user preference positive value for relevant document and relevant value for negative document.

3 Web Track Experiment

3.1 Web methodology

We tested the relevance backpropagation strategy using a user preference. In fact, we consider that the relevance of the documents are not all the same, but depends on the user satisfaction. In Mercure system, the user satisfaction is represented by the $coef\_rel$ parameter assigned by the user. Because the user does not judge the document, in the pseudo relevance backpropagation, the top retrieved documents are assumed as relevant. The "user" preference is then assigned to a document according to its rank in the retrieved set. For the trec9 experiment, the top 12 documents were assumed to be relevant. The user preference is computed as follows :

- $coef\_rel = 1$ for the documents ranked from 1 to 5
- $coef\_rel = .75$ for the documents ranked from 6 to 10
- $coef\_rel = .5$ for the documents ranked from 11 to 12

3.2 Results

Four runs based on content only were submitted:

1. Mer9Wt0: simple search using the title field
2. Mer9Wt1: title field + pseudo-relevance feedback based on Mercure relevance backpropagation. The top 12 retrieved documents are assumed to be relevant and the $coef\_rel = 1$
3. Mer9WtMr: title field + pseudo-relevance feedback based on Mercure relevance backpropagation. The top 12 retrieved documents are assumed to be relevant and the $coef\_rel$ are computed using the user preferences as described above.
4. Mer9Wtnd: simple search using the title+Description+Narrative fields.

Unfortunately, the official results were wrong because of a mistake in our indexing script. This explains in part the bad results obtained this year by Mercure comparing to those obtained in the previous TREC. This will be modified and the corrected results will be integrated in the final paper. Consequently to the problem encountered with the indexing process, the experiments carried out by taking hyperlinks into account could not be done in time and will be included in the final paper too.

Type	Run	average precision	Exact precision
Mer9Wt0	title simple search	0.0996	0.1274
Mer9Wt1	title +simple pseudo-rb	0.0114	0.0242
Mer9WtMr	title+simple pseudo-rb basen user preference	0.0154	0.0307
Mer9WtMr	TDN fields +simple serach	0.0140	0.0295

Table 1: Web component results - 50 queries

4 Batch and Routing Experiments

The batch and routing experiments were performed using Mercure system. The profiles were learned using the same learning algorithm as before: the relevance backpropagation. The relevance value assigned to each document was used as user preference (2 and 1). It corresponds to $Coef\_rel$ in the relevance backpropagation algorithm.
The filtering algorithm starts with an initial query, built from all the topic parts, and its OHSU87 relevance judgements. A pool of queries based on the learning method was then selected. The OHSU88-91 documents were used as test data.

4.1 Batch Filtering

The profiles in the batch task are learned using the relevance backpropagation method. Two techniques are tested to be compared. The first one corresponds to filtering documents using learned (positive, relevant) profiles. The second represents filtering documents using relevant and non-relevant (negative) profiles. These methods are detailed below. We use the phrase *"positive profiles"* for relevant learned profiles, and *"negative profiles"* for non-relevant profiles. The TREC standard output file of each query was analyzed to build an output file containing:

$$< topic >< func >< value >< thresh >< rank >< prec >< recall >< method >$$

In this section, we detail both the batch filtering methods and the results.

4.1.1 Batch filtering using positive profiles

As it has been done in [7]. The document activation weights which maximizes the utility function were found and selected as thresholds. Then the queries corresponding to these thresholds were selected and tested on a set of test documents. The documents weighted by values higher than the threshold were selected for the corresponding query.
In the first run (mer9b1), we build profiles by learning using relevance backpropagation on the training dataset, then we apply them on the test dataset.
The following algorithm is used:

```
For each profile $P$
  1. evaluate $P$ on the training dataset
  2. select top 1000, let $result_0$ be the obtained document ranked list
  3. $i = 1$
  4. repeat
     (a) build a new profile $P_i$ by relevance backpropagation
     (b) evaluate $P_i$ on the training dataset
     (c) select top 1000, let $result_i$ be the obtained document ranked list
     (d) inc $i$
```

until $i = max\_iteration$

5. for each $r \in \{1 \ldots 1000\}$, $i \in \{0, 1, \ldots max\_iteration\}$
    (a) $result_{ir}$ contains top r documents from $result_i$
    (b) evaluate $result_{ir}$ on $utility\_[T9U]$
6. select profile P_i such as $\exists r \in \{0, 1 \ldots max\_iteration\}$ where $result_{ir}$ gives the best $utility\_[T9U]$ value
7. apply P_i on the test dataset, $test\_result_i$ is the obtained document ranked list
8. $submitted$ contains documents in $test\_result_i$ having a rsv at least equal to the rsv of the r^{th} document
9. submit the selected list $submitted$

In the experiments, we carried out relevance backpropagation twice. We found that this number of iterations was enough to learn the profile.
We computed the utility on the top 1000 documents only as this set is likely to contain most of the relevant documents.

4.1.2 Batch filtering using positive and negative profiles

The second run (mer9b2) is based on negative profile building. We built -in addition to the positive profile- a negative profile and applied both positive and negative profiles on the test dataset as detailed below.
A negative profile is a profile aiming at excluding non-relevant documents from the top ranked ones. A document is filtered when it is potentially relevant compared to the positive profile, and non-relevant compared to the negative profile. Generally, negative profile is not provided by the user, but the system can build it by learning. It is built starting from the non-relevant selected documents.
In our experiments, negative profiles are built by relevance backpropagation. We select the top 50 non-relevant documents from the filtered ones resulting from training documents evaluation. These documents are used as relevant documents in a relevance backpropagation process, the resulting learned profile is considered as the negative profile. As there is no initial negative profile, we use different values of Ma and Mb: we assign 0 to Ma and 1 to Mb and thus, the weight assigned to each term in the negative profile is its output activation done by this relevance backpropagation process.
Two values are then learned using the training data set, a and b, used in filtering documents using positive and negative profiles. a is a multiplicative value of the relevance threshold, it allows to select more documents, so it is less than 1. b allows to select documents which [positive profile rsv/negative profile rsv] ratio is higher than b, so it defines [negative rsv/positive rsv] ratio threshold. Batch filtering document process using negative profiles is described as follows:

For each document D_i

1. compute an rsv value for the positive profile: $rsv_p(D_i)$

2. if $rsv_p(D_i) > a * threshold$

   (a) compute an rsv value to the negative profile: $rsv_n(D_i)$

   (b) if $rsv_p(D_i) > b * rsv_n(D_i)$ filter document D_i else reject document D_i

   else reject document D_i

We varied a in an interval raised by 1 and b in an interval undervalued by 1, and retained the values giving the best utility-[$T9U$] value. a should be less than 1 in order to select more documents, and b is more than 1 to filter documents which are most likely to be relevant to the positive profile and non-relevant to the negative profile. This algorithm was processed on the test documents resulting from mer9b1 run. Note that if $a = 1$ and $b = 0$, *mer9b1* and *mer9b2* are the same.

In our experiments, we selected as positive profiles the profiles applied to *mer9b1* run, and carried out relevance backpropagation using top 50 non-relevant documents from the document ranked list resulting from evaluating these profiles on the training dataset. a and b values were then learned on the dataset. Finally, positive, negative profiles, rsv threshold, a and b were applied on the test dataset.

4.1.3 Batch filtering results

Table 2 lists the comparative batch results on utility-[$T9U$].

TREC batch filtering			
Run	$\geq median$	$= max$	$score = 0$
mer9b1	38	10	5
mer9b2	22	5	0

Table 2: Comparative batch filtering results

It can be seen that in both mer9b1 and mer9b2 runs the results are quite good, 13 best queries in the first run and 7 in the second run. For 5 queries, no documents were retrieved using positive profiles only, whereas documents are retrieved for these queries using positive and negative profiles.

Batch experiments have been performed to show the effectiveness of using negative profiles. Despite better results using positive profiles only, the first results using negative profiles are promising. Using negative profiles on the training documents improve results for many values of a and b. The best values were chosen and applied on the test documents. Future work will be devoted to taking into account other parameters in order to obtain significant results when using negative profiles.

4.2 Routing task

We experiment two methods in the routing track:

- using a similar method then in the batch filtering track

- using the results of a factorial correspondence analysis applied to a sub-collection

In the first method, the queries having the best average precision in the training dataset were selected as routing queries, and are applied on the test documents.

The second method we experiment expands each query using the training document set before it is sent to Mercure. To expand the query, we first performed a search on the training document set and selected the top ranked documents. Then a list of the most representative terms -either single terms or phrases- is extracted from this sub-collections. Once selected, the list of terms is used to build a crossing matrix that crosses the terms and the documents: $M = [g_i * l_{ij}]$ where l_{ij} is a local weight that reflect the importance of term i within document j and g_i is a global weight that reflect the importance of term i within the sub-collection of documents. A Correspondence Factorial Analysis (CFA) [1][2] is then performed on the matrix and the relevant factorial axes are selected. A factorial axis is considered as relevant if it contributes to relevant documents. The terms are then re-ranked according to their contribution to the selected factorial axes. The top ranked terms are used to expand the query after altering their weight using the cosines of the angles between them and the relevant documents within the factorial space.

The main purpose of this method is to evaluate the use of the CFA to find out the dimensions -parts- of factorial space that are characterized by relevant documents and then to find what terms contribute most to those dimensions.

4.2.1 Extraction of the sub-collection

CFA is about orthogonal factorization and singular value decomposition of a square matrix which dimension is the collection document number. Almost all the algorithms used to achieve these factorization and decomposition are iterative and very time consuming. The overall time they take mainly depends on the matrix dimension. For this reason we chose to perform the analysis on a sub-collection rather than on the entire training collection. The sub-collection contains top ranked documents returned by a first search using the *Vigie*[5] system. This system includes the stop word removal and the Porter stemming.

4.2.2 Items extraction

In order to select terms, we use the term weight presented in the OKAPI system [8]
$w_{ij} = \frac{tf * log(\frac{N-n+0.5}{n+0.5})}{2*(0.25+0.75*(\frac{dl}{avdl})+tf}$
where:
N is the collection document number
n is the number of documents containing the term i
tf is the term frequency within the document j
dl is the document length -size-
avdl is the average document size
the weights are calculated for each pair of term/document and each term is associated to its second highest associated weight so that we can rank the terms and select the top of them. This means that the selected terms are representative of at least two documents.

4.2.3 Terms re-ranking strategy and selection

The principal goal of the method is to expand the topic -query- by adding new terms and modifying the weights of the existing ones. The terms added are the ones that contribute the most to the factorial axes related to relevant documents. In other words the axes to which the relevant documents contribute highly. The top ranked terms are used to expand the query. Using this method, a term can be added more than once. The new query is constituted of new terms and already existing terms.
The weight of new terms is calculated as bellow: $w_i = K * p_i * Max_j cos(T_i, D_j)$
The weight of the already existing is calculated as bellow:
$w_i = w_{i,old} * K * p_i * Max_j cos(T_i, D_j)$
K is a constant that depends on the number of terms and documents,
p_i is the number of time term T_i is selected to added to the topic,
$w_{i,old}$ is the initial weight of term T_i,
(T_i, D_j) is the angle formed by relevant document D_j and term T_i within the factorial space.

4.2.4 Routing results

Table 3 shows the routing results at average precision.

TREC Routing			
run	$= max$	$\geq median$	AvgP
mer9r1	6	38	0.235
mer9r2	0	20	0.185

Table 3: Comparative routing results at average precision

In average, mer9r1 obtained better precision results than mer9r2.

The average precision obtained in mer9r1 is slightly lower than the average precision obtained in TREC8. It can be seen that using the relevance values as user preferences improves routing results. Thus results could probably be improved more by assigning different relevance values depending on documents.
The results of 10 of the queries that obtained equal or better results than the average using the second method (run mer9r2) are higher than the one of run mer9r1. The combination of the two methods has to be explored further.

References

[1] J. P. BENZÉCRI *Correspondence Analysis Handbook*, MARCEL DEKKER ED., NEW YORK, 1992.

[2] M. W. BERRY, Z. DRMAC, E. R. JESSUP *Matrices, Vector Sparces, and Information Retrieval* IN SIAM REVIEW VOL. 41. NO 2, PP 335-362, 1999.

[3] M. Boughanem, C. Chrisment & C. Soule-Dupuy, *Query modification based on relevance backpropagation in Adhoc environment*, Information Processing and Management, 35(1999)121-139 April 1999.

[4] M. Boughanem, T. Dkaki, J. Mothe & C. Soule-Dupuy, *Mercure at trec7*, Proceedings of the 7th International Conference on Text REtrieval TREC7, E. M. Voorhees and Harman D.K. (Ed.), NIST SP 500-236, November 1998.

[5] T. Dkaki, B. Dousset, J. Mothe *Analyse d'informations issues du Web avec Tétralogie* In Proceedings of the Veille Stratégique scientifique et technologique Conférence, pp 159-170, 1998.

[6] K. Hoashi, K. Matsumoto, N. Inoue, K. Hashimoto *Document filtering method using non-relevant information profile* Proceedings of the ACM/SIGIR, Athens,Greece July 2000

[7] S. Robertson and al *Okapi at TREC-6*, Proceedings of the 6th International Conference on Text REtrieval TREC6, Harman D.K. (Ed.), NIST SP 500-236, November 1997.

[8] S. E. Robertson, S. Walker *Okapi/Keenbow at TREC-8* In Proceedings of the TREC 8 Conference, National Institute of Standards and Technology, 1999.

The HAIRCUT System at TREC-9

Paul McNamee, James Mayfield, and Christine Piatko
The Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Road
Laurel, MD 20723-6099 USA
Paul.McNamee@jhuapl.edu
James.Mayfield@jhuapl.edu
Christine.Piatko@jhuapl.edu

Overview

The Hopkins Automated Information Retriever for Combing Unstructured Text (HAIRCUT) is a research IR system developed at the Johns Hopkins University Applied Physics Laboratory (JHU/APL). HAIRCUT benefits from a basic design decision to support flexibility throughout the system. One specific example of this is the way we represent documents and queries; words, stemmed words, character n-grams, multiword phrases are all supported as indexing terms. This year we concentrated our efforts on two of the tasks in TREC-9, the main web task and cross-language retrieval in Chinese and English.

Small Web Task

For this task we indexed documents using two types of indexing terms, unstemmed words and character n-grams using n=6. Summary information of the two indices is shown in Table 1. The difference in the number of documents is likely attributable to a few documents that contain a single short word from which no six character sequence can be formed. Note that the use of 6-grams greatly increased both the size of the dictionary and the size of the index files. No attempt was made compress our data structures and reduce the amount of disk space required although such techniques have been successful with both words [12] and n-grams [10].

Each document was processed in the following fashion. First, we ignored HTML tags and used them only to delimit portions of text. Thus no special treatment was given for sectional tags such as `<TITLE>` or `<H1>` and both tags and their attribute values were eliminated from the token stream. The text was lowercased, punctuation was removed, and diacritical marks were retained. Tokens containing digits were preserved; however only the first two of a sequence of digits were retained (e.g., 1920 became 19##). The result is a stream of blank-separated words.

When using n-grams we construct indexing terms from the same sequence of words. These n-grams may span word boundaries; an attempt is made to discover sentence boundaries so that n-grams spanning sentence boundaries are not recorded. Thus n-grams with leading, central, or trailing spaces are formed at word boundaries.

Queries were parsed in the same fashion as were documents with two exceptions. On some of our title only runs we attempted to correct the spelling of words that did not occur in our dictionary. Also, we tried to remove stop structure from the description and narrative sections of the queries using a list of about 1000 phrases constructed from past TREC topic statements.

	# docs	# terms	index size
words	1,588,374	3,019,547	2.96 GB
6-grams	1,588,169	19,209,934	36.0 GB

Table 1. Index statistics for the wt10g collection

In all our experiments we used a linguistically motivated probabilistic model. This model, described in a report by Hiemstra and de Vries [2], is essentially the same model that was used by BBN in TREC-7 [9]. The similarity calculation that is performed is:

$$Sim(q,d) = \prod_{t=terms} \left(\alpha \cdot f(t,d) + (1-\alpha) \cdot df(t)\right)^{f(t,q)}$$

Equation 1. Similarity calculation.

where *f(t,d)* is the frequency of term *t* in document *d* and *df(t)* denotes the document frequency of *t*.

After the query is parsed each term is weighted by the query term frequency and an initial retrieval is performed followed by a single round of relevance feedback.

To perform relevance feedback we first retrieve the top 1000 documents. We use the top 20 documents for positive feedback and the bottom 75 documents for negative feedback; however duplicate or near-duplicate documents are removed from these sets. We then select terms for the expanded query. After retrieval using this expanded and reweighted query, we have found a slight improvement by penalizing document scores for documents missing many highly ranked query terms. We multiply document scores by a penalty factor:

$$PF = 1.0 - \left(\frac{\#\text{ of missing terms}}{\text{total number of terms in query}} \right)^{1.25}$$

Equation 2. Penalty function for missing terms.

As can be seen in Table 2, we use only about one-fifth of the terms of the expanded query for this penalty function

	# Expansion Terms	# Penalty terms
words	60	12
6-grams	400	75

Table 2. Number of expansion terms and penalty terms by indexing scheme.

Several of our official runs were formed by merging baseline ranked lists of documents, for example, merging a word-based query and a 6-gram based query. We merged separate ranked lists by first normalizing document scores and then linearly combining values from different runs, an approach that was successful for us in TREC-8 [7].

We conducted our work on a 4-node Sun Microsystems Ultra Enterprise 450 server. The workstation had 2.5 GB of physical memory and access to 100 GB of dedicated hard disk space.

Official Results

For the most part we ignored the web-nature of the documents and relied on textual content to rank documents. We did however, try two techniques to boost our content-based runs. Both techniques were motivated by the track guidelines. First, we attempted to exploit hyperlink structure and submitted two runs that used backlink frequency to rerank content-based runs. Secondly, we attempted to correct misspellings in title-only queries.

We submitted six official submissions in the small web track, four of the runs were solely based on document content and the other two were an attempt to utilize backlink frequency information to improve a content-based run.

Three of our four content-based runs differ only in the selection of which parts of the topic statements were used. Thus *apl9t*, *apl9td*, and *apl9tdn* used the title, title and description, and title, description, and narrative sections, respectively. The fourth run, *apl9all* was a combination of the three other runs. A summary of each run's performance on the task is shown in Table 3.

	avg prec	recall	# best	# ≥ median
apl9t	0.1272	1276	0	28
apl9td	0.1917	1535	2	33
apl9tdn	0.1785	1584	1	32
apl9all	0.1948	1609	0	37

Table 3. Content-based runs for the Small Web task.

We were surprised by lower than expected results in the web task. During brief post-hoc analysis of our constituent runs we observed that relevance feedback had an adverse effect on our runs; rather than the 25-30% increase in average precision that we typically find, average precision decreased by roughly 10%. It will require further analysis to discover the cause for this phenomenon. We observe that the mean number of relevant documents per query, 52.3, is lower than past ad hoc TREC tracks and it is possible that this would reduce the benefit normally associated with automated relevance feedback.

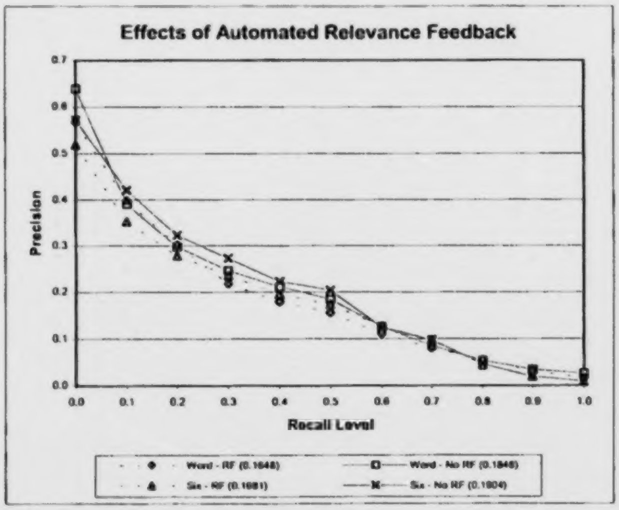


Figure 1. Adverse effects of blind relevance feedback.

Naïve Use of Backlink Frequency

We made a simple attempt to incorporate link frequencies in our results. This was done in a very simple way - we multiplied a document's score in a content-based retrieval by a multiplicative factor derived from backlink frequency and resorted the retrieved documents. The exact computation was:

$$BLFactor(d) = 0.1 + 0.9\sqrt{\frac{backlinkcount(d)}{MaxBacklinkCount}}$$

Equation 3. *MaxBacklinkCount* is the number of documents that link to the most linked-to document.

Comparing the results in Table 3 and Table 4, it is clear that such a simple attempt to exploit backlink counts is insufficient.

	avg prec	change	# best	# ≥ median
apl9lt	0.1062	- 0.0210	0	25
apl9ltdn	0.1494	- 0.0454	0	26

Table 4. Link-influenced runs corresponding to *apl9t* and *apl9tdn*.

Use of Spelling Correction

If three or fewer documents in the TREC-8 collection contained a topic term, we attempted spelling correction on that term. First, we looked for words occurring in at least five documents that were one insertion, deletion, substitution, or transposition away from the misspelled word. If such a word was found, we used it in lieu of the misspelled word; if more than one such word was found, we selected the one that occurred most frequently (this led us to correct 'tartin' to 'martin' rather than 'tartan'). If no correction was found, we then tried to split the word into two pieces of three characters or more, each of which appeared in at least five TREC-8 documents. If no such pair was found, we left the word uncorrected.

The results of our attempts at spelling correction are shown in the following table:

Topic	Original	avg prec	Correction	avg prec	Change
463	tartin	0.0000	martin	0.0000	0.0000
464	nativityscenes	0.0000	nativity scenes	0.0000	0.0000
474	bennefits	0.0003	benefits	0.0002	-0.0001
476	aniston	0.1517	anniston	0.0062	-0.14.55
483	rosebowl	0.0108	rose bowl	0.3198	+0.3090
487	angioplast7	0.0000	angioplasty7	0.1553	+0.1553

Table 5. Impact of spelling correction.

These results reflect word-based title-only runs with relevance feedback. Spelling correction helped us dramatically on two queries, and hurt us on one.

Cross-Language Task

The TREC-9 CLIR task consisted of bilingual retrieval of Chinese newspaper articles from English queries. A monolingual Chinese-Chinese run was also permitted. This was JHU/APL's first experience with Chinese document retrieval and we learned quite a lot from the experience. Undaunted by our inability to read Chinese, we attempted the task with only an English/Chinese parallel corpus and a minimal knowledge of the Big-5 encoding. Our CLIR experiments focused on two questions, namely, "How do 2- and 3-grams compare as indexing terms in unsegmented Chinese text?" and "Does query translation with parallel corpora perform on par with an available machine translation system?"

Philosophically, we desire to maximize cross-language performance using few language-specific resources. Although segmenters and dictionaries are available for a high-density language such as Chinese, many languages lack these tools. Additionally such resources are rarely in a standard format and the quality of the resource depends greatly on the source.

Though we did perform an experiment indexing only the raw bytes of the collection, on the whole it seemed better to process the Big-5 encoded documents on a character basis. The CJKV text by Ken Lunde was an invaluable aid in our software development [6]. We did not segment the text, and instead elected to index the documents using both 2- and 3-grams. Nie and Ren have previously reported that 2-grams perform comparably with words on the TREC 5/6 Chinese collection and that a combination of both is best [11]. We wanted to assess the use of 3-grams in a straight-up comparison with 2-grams.

We tried translating the topic statements in three different ways, two using a parallel corpus and one using an online machine translation tool. In our monolingual Chinese run we attempted to remove stop structure using translations of our English stop phrases. We used the same linguistically motivated probabilistic model that was used for our English web retrieval. Most of our official runs were produced by combining individual runs using both 2- and 3-grams, an approach that as it turns out, depressed our results.

	# docs	# terms	index size
2-grams	127938	1974077	673 MB
3-grams	127938	15185076	959 MB

Table 6. Index statistics for the TREC-9 Chinese collection.

Translation Using Hong Kong Parallel Corpora

About one month before the CLIR results were due at NIST we observed that we had no in-house method for translating English to Chinese. We quickly obtained two parallel English/Chinese collections from the Linguistic Data Consortium (LDC), the Hong Kong Laws Parallel Text collection [4] and the Hong Kong News Parallel Text collection [5].

The Laws collection contains roughly 310,000 aligned sentences. The News collection contains roughly 18,000 aligned documents. Both collections are encoded in Big-5 which matches the encoding in the TREC-9 Chinese collection.

We built a hybrid collection from the Laws collection and from aligned sections of the News documents. We indexed the collection twice, both with 2-grams and 3-grams. Summary information about these two indices is shown in the following table:

	# docs	# terms	index size
English words	344,299	46,951	105 MB
Chinese 2-grams	343,714	553,358	195 MB
Chinese 3-grams	333,007	2,908,676	270 MB

Table 7. Statistics for APL's hybrid parallel collection.

Official results

We submitted four official runs for the CLIR task, *apl9xmon*, *apl9xtop*, *apl9xwrd*, and *apl9xcmb*, that are described below. Each run is produced by combining multiple base runs. All of the base runs made use of relevance feedback. The number of expansion terms varied depending on the indexing terms; 100 expansion terms were used with the 2-gram index and 400 terms were used with 3-grams.

Our only monolingual submission was *apl9xmon*. This run was produced by combining six base runs, title-only, title + description, and title + description + narrative, using both 2- and 3-grams.

Our first method for query translation followed the approach we used successfully in the CLEF-2000 evaluation [8], namely, pre-translation expansion using highly ranked documents from a document collection in the same language as the source query followed by individual term translation using our parallel collection. Using this approach, the run, *apl9xtop*, was built from two base runs that were produced from 2- and 3-grams. The base runs used queries produced by expanding full topics from documents in the TREC-8 collection.

We were concerned that using the TREC-8 collection as an expansion collection might not be a good idea since it is not contemporaneous with the Chinese collection. We therefore tried a word-by-word translation of the topic statements, also using the parallel collection. The run *apl9xwrd* was produced by combining six base runs (2-, 3-grams; T, TD, TDN queries).

The final run, *apl9xcmb*, was simply a combination of all base runs used in *apl9xtop*, *apl9xwrd*, and the unofficial machine translation run, *apl9xibm*.

	avg prec	recall	# best	# ≥ median	% mono
apl9xmon	0.3085	621	5	20	100 %
apl9xtop	0.0763	360	0	7	24.7%
apl9xwrd	0.1076	416	0	8	34.9%
apl9xcmb	0.1523	535	0	11	49.4%

Table 8. Official results for CLIR task

We wanted to compare translation using our parallel collection to available machine translation. We were not in possession of Chinese MT software in-house so we relied on a web-based translation. The first operational web-based translation service we found was the IBM AlphaWorks server [3]. We had no previous experience with this service or knowledge of its methods or quality; we decided to use it solely based on convenience. The unofficial run, *apl9xibm* was produced from six base runs (2-, 3-grams; T, TD, TDN queries).

Comparing 2-grams and 3-grams

Our decision to submit combined runs using both 2- and 3-grams was based on experience that shows benefit from a combination of multiple, reasonable quality results. As it turns out, our runs using 3-grams performed appreciably worse than those using 2-grams. Average precision and recall for the monolingual base runs used in *apl9xmon* are shown in Table 9.

It seems clear that 2-grams are preferable to 3-grams, at least on a collection of this size. This trend seems to hold both in monolingual retrieval with natural language queries and in bilingual retrieval using word-based 'translations'. We created a post-hoc monolingual run using only the 2-grams and saw average precision increase from 0.3085 in *apl9xmon* to 0.3339, an 8.2% increase.

		avg prec	recall
2-grams	T	0.2926	606
	TD	0.3154	622
	TDN	0.3333	624
3-grams	T	0.1991	572
	TD	0.2170	571
	TDN	0.2368	555

Table 9. Comparing 2- and 3-grams using monolingual queries.

A previous study by Chen et. al. [1], examined the relative merits of 1-, 2-, and 3-grams (as well as several other methods of indexing) using the TREC-5 Chinese collection. Though the data, character encoding, and retrieval model differ from this present study, the relative performance between 2-grams and 3-grams is quite similar for several metrics. On automatic long queries they report average precision of 0.3677 for 2-grams and 0.2405 for 3-grams, a performance ratio of 1.529; from values in Table 9 we compute a comparable ratio of 1.408. Looking at relevant documents retrieved we report a ratio of 1.123 to their 1.162.

Performance of Different Translation Schemes

Another thing we wanted to examine was the effect of using different query translation methods. Our three methods achieved similar performance. Rather than compare the combined runs, we instead look at the constituent base runs. The following tables reveal the performance achieved by each run and its relative performance to *apl9xmon*. For each strategy the best performance was observed when 2-grams were used on full-length topic statements.

		avg prec	recall	% mono
2-grams	TDN	0.1175	341	38.1%
3-grams	TDN	0.0261	237	8.46%

Table 10. Bilingual results using pre-translation expansion (topic expansion)

		avg prec	recall	% mono
2-grams	T	0.1036	409	33.6 %
	TD	0.1214	455	39.3 %
	TDN	0.1261	461	40.9%
3-grams	T	0.0464	254	15.0%
	TD	0.0440	309	14.3%
	TDN	0.0245	244	7.94%

Table 11. Bilingual results using individual word translation

		avg prec	recall	% mono
2-grams	T	0.0674	385	21.8%
	TD	0.1017	487	33.0%
	TDN	0.1284	517	41.6%
3-grams	T	0.0512	305	16.6%
	TD	0.0774	335	25.1%
	TDN	0.0773	374	25.1%
apl9xibm		0.1000	497	32.4%

Table 12. Bilingual results using IBM's AlphaWorks Translator

The performance achieved by each of the translation methods was very similar. The precision-recall graph in Figure 2 shows the performance of each query translation scheme using 2-gram indexing and full topic statements. The graph shows that while the average precision using each method is nearly the same, the AlphaWorks translator performs slightly better at the high-precision part of the curve.

None of the bilingual runs achieves comparable performance to the monolingual run and our best official bilingual submission, *aplxcmb* only achieves performance of 49.4% of our official monolingual run, *apl9xmon*. This is significantly lower percentage than the 70-80% we obtained in our experiments with the CLEF-2000 workshop that was devoted to European languages [8].

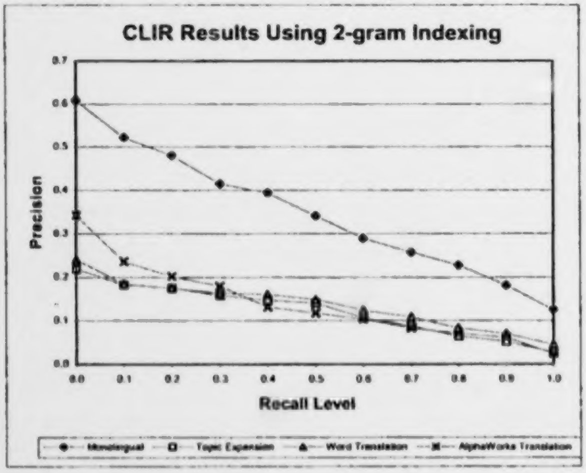


Figure 2. Precision-recall curve for CLIR runs

See the following page for an example of the query translations we used.

Topic CH73

Official English Query

<title> AIDS in China

<desc> Description:
Find documents that report on the number of cases of AIDS in China, the names and locations of AIDS research and treatment facilities in China, and the number of deaths per year attributed to AIDS in China.

<narr> Narrative:
Documents that quote specific total numbers or percentages for people diagnosed with AIDS in China are relevant. Documents containing the official names and/or locations of China's research and treatment facilities are relevant. Documents revealing China's total number of fatalities per year due to AIDS are relevant.

IBM AlphaWorks Translation

<title> 在中國

<desc>描繪中有幫助:爲檔案找到那份關於在中國中的愛滋病的情況的數目.名字和位置愛滋病研究和處理設施在中國中的,和每被把年歸於在中國中的愛滋病的死的數目的報告 .

<narr>敘事:爲用在中國中的愛滋病被診斷的人們引用特有總數或者百分比的檔案是有關.含有官方名字和或位置中國的研究和處理設施的的檔案是有關.暴露中國的總每由於愛滋病年的死亡的數目的檔案是有關.

English word	Top 2-gram	Top 3-gram
aids	愛滋	愛滋病
china	中華	華人民
cases	況下	情況下
research	研究	的研究
number	數目	的數目
treatment	治療	熱處理
hiv	滋病	愛滋病
deaths	生死	生死登
total	的總	不得超
diagnosed	診斷	生署
prevention	防止	《防止
health	生	生主
official	產管	產管理
chinese	英文	中英文
numbers	號碼	附:
carriers	散貨	／散貨
infected	感染	受感染
provinces	轄市	直轄市
intravenous	二年	大專新
disease	疾病	傳染病
county	縣級	縣級以
beijing	北京	在北京
education	教育	教育
reclassification	none	none
virus	病毒	病毒
spread	蔓延	生署
patient	病人	生署
adolescents	青少	青少年
dept	總協	理受影
yunnan	雲南	雲南省
tracy	none	none
ivdu	none	none
mainland	內地	國內地
infection	感染	生署
indicates	顯示	本標誌
foreigners	外籍	外籍人
regions	地區	自治區
spreading	擴散	（星期
risk	風險	的風險
reported	星期	（星期
publicity	宣傳	傳活動
angeles	洛杉	洛杉磯
adults	成人	成年人
los	洛杉	洛杉磯
characteristics	特徵	統計調
facilities	設施	的設施
drug	藥物	危險藥
monitoring	監察	（星期
thomas	星期	日（星
medical	醫生	間醫生
negative	負面	影響
control	控制	或控制
table	省暨	交立法
discovered	發現	關公布
ministry	交部	外交部
causes	導致	或導致
december	二月	2 月
cities	城市	香港的
chen	教授	（星期
minzhang	none	none

Figure 3. Two query translation methods are compared. The original English version of topic CH73 is shown along with the results of the IBM AlphaWorks translator. In the table on the right the query used in apl9xtop is partially displayed. The first column contains the best sixty terms produced by searching the TREC-8 ad hoc English documents using the official English version of topic CH73. The second column contains the top-ranked 2-gram extracted from our parallel collection; the third column contains the top-ranked 3-gram. During retrieval the top three 2-grams and the top 10 3-grams were used; however, only the top term is shown here due to space constraints.

BEST COPY AVAILABLE

Conclusions

This year we participated in two tracks that each presented new challenges.

In the small web task, we focused on content-based methods and tried two techniques to 'accommodate' the web-nature of the task. The first technique was a rudimentary use of backlink counts that proved too simplistic to be beneficial. The second technique, spell correcting misspelled short queries was generally beneficial, however it backfired in certain instances. We found automated relevance feedback to have a deleterious effect on our performance, a finding that warrants further investigation.

Though our team is experienced in cross-language retrieval, we had no experience in Asian language retrieval. We started the Chinese task with no ability to read Chinese and no language resources such as segmenters or dictionaries to draw on. Due to time constraints we were unable to make use of the TREC-5/6 training data and thus we entered the task relatively unprepared. We relied on our general experience using n-grams as indexing terms, a quickly acquired knowledge of the Big-5 encoding, and an English/Chinese parallel collection.

From our experience in the CLIR track we draw the following lessons. First, 2-grams are preferable to 3-grams for indexing Chinese. We remain open to the possibility that other techniques may be better still – for example, using both 2-grams and 3-grams, or 2-grams and segmented words. Our second observation is that corpus-based translation is a viable alternative to extant machine translation software. However, our present results in English to Chinese, bilingual retrieval seem to fall well short of Chinese monolingual retrieval. Now that we have some experience in Chinese text retrieval and a training collection to draw from, we will endeavor to refine our methods to narrow this gap.

References

[1] A. Chen, J. He, L. Xu, F. C. Gey, and J. Meggs, 'Chinese Text Retrieval Without Using a Dictionary'. In the Proceedings of the 20th International Conference on Research and Development in Information Retrieval (SIGIR-97), pp. 42-49, July 1997.

[2] D. Hiemstra and A. de Vries, 'Relating the new language models of information retrieval to the traditional retrieval models.' CTIT Technical Report TR-CTIT-00-09, May 2000.

[3] IBM's AlphaWorks Translation Service, http://www.alphaworks.ibm.com/aw.nsf/html/mt

[4] Linguistic Data Consortium (LDC), Hong Kong Laws Parallel Text, described at http://www.ldc.upenn.edu/Catalog/LDC2000T47.html

[5] Linguistic Data Consortium (LDC), Hong Kong News Parallel Text, described at http://www.ldc.upenn/Catalog/LDC2000T46.html

[6] Ken Lunde, *CJKV Information Processing*, O'Reilly & Associates, January 1999.

[7] J. Mayfield, P. McNamee, and C. Piatko, 'The JHU/APL HAIRCUT System at TREC-8.' In E. M. Voorhees and D. K. Harman, eds., *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*. To appear.

[8] P. McNamee, J. Mayfield, and C. Piatko, 'A Language-Independent Approach to European Text Retrieval.' Draft version in the *Working Notes of the CLEF-2000 Workshop*, Lisbon, Portugal, September 2000.

[9] D. R. H. Miller, T. Leek, and R. M. Schwartz, 'A Hidden Markov Model Information Retrieval System.' In the Proceedings of the 22nd International Conference on Research and Development in Information Retrieval (SIGIR-99), pp. 214-221, August 1999.

[10] E. Miller, D. Shen, J. Liu, and C. Nicholas, 'Performance and Scalability of a Large-Scale N-gram Based Information Retrieval System.' In the *Journal of Digital Information*, 1(5), January 2000.

[11] J.-Y. Nie and F. Ren, 'Chinese Information Retrieval: using characters or words?'. In *Information Processing and Management*, 35(4), 1999.

[12] I. Witten, A. Moffat, and T. Bell, Managing Gigabytes: Compressing and Indexing Documents and Images 2nd edition, Academic Press, 1999.

Reflections on "Aboutness"
TREC-9 Evaluation Experiments at Justsystem

Sumio FUJITA
JUSTSYSTEM Corporation
Brains park, Tokushima, 771-0189 JAPAN
+81-88-666-1000
Sumio_Fujita@justsystem.co.jp

ABSTRACT

TREC-9 evaluation experiments at the Justsystem site are described with a focus on "aboutness" based approach in text retrieval.

Experiments on the effects of supplemental noun phrase indexing, pseudo-relevance feedback and reference database feedback in view of the effect of various length of queries are reported.

The results show that pesudo-relevance feedback is always effective while reference database feedback is effective only with very short queries.

We reconfirmed that supplemental phrasal indexing is more effective with longer queries.

Keywords

Aboutness, Supplemental Phrasal indexing, phrasal terms, pseudo-relevance feedback, reference database, vector space model.

1. INTRODUCTION

Automatic indexing of modern information retrieval systems typically adopts bag-of-word representation, in which each word is considered as a dimension of the vector representing an information item, as internal representation of "aboutness". It is well known that such simple representation usually performs, as well as, if not better than, some more sophisticated ones according to empirical evaluations.

Grammatical relations or functional words are normally considered as neutral in view of thematic discrimination of text documents. On the other hand, content words(or lexemes, if we need to be more attentive for linguistic terminology) are semiologically meaningful units in language systems which refer to conceptual/substantial entities or relations in the subject domain described by the documents. It is plausible that the author of documents and the user submitting search requests share the same terminology when describing the subject concept in question either in their documents or in queries. The notion of "aboutness" is considered as a set of terms evoking a subject concept, which is hopefully shared by many people including authors, indexers and users of the system.

2. "ABOUTNESS"

The concept of "aboutness" plays an essential role in modern information retrieval technologies where "author's aboutness" [Ingwersen 93] is extracted automatically from text documents by automatic indexing procedures.

2.1 "Aboutness" as Representation of Information Objects

The basic hypothesis behind our TREC-9 strategies is that the "aboutness" of a subject topic consists of "foreground" part and "background" part and terms belong to either one of them. This distinction is inspired by the metaphor of "aboutness" of visual information items. People are clearly distinguishing foreground images from background ones when talking about "aboutness" of for example picture images. A foreground image might be a person or some objects located in the center of the picture and constitute the motif of the picture. Background images can help to identify the scene where the motif image is located and sometimes clue images are hidden in background when some implicit information is given in the picture.

In text retrieval, we can consider concepts that directly related to the motif as foreground and concepts that simply constitute the scene of the motif as background.

The term weighting should accordingly take this into consideration so that the terms that belong to "foreground aboutness" should be more weighted than "background aboutness".

Foreground terms are mainly extracted from <title> or <description> fields of topic description.

A stratified automatic feedback strategy is adopted in order to extract mainly terms of "background aboutness" both from the target document database(wt10g) and a reference database(TREC CD4&5).

2.2 Single words as a minimum unit of "aboutness"

Single words are indexed as basic units of "aboutness" but also noun phrases are extracted as supplemental indexing units.

For example, from the TREC topic 468 the following terms are extracted:

PH(incandescent light bulb)
PH(incandescent light),PH(light bulb)
incandescent, light, bulb

Longer phrases have normally more specific reference consequently they seem to focus more on foreground part of subject description while a set of constituent single word terms are referring to the subject as if it is on background.

Changing relative weighting of phrases against single word terms, "aboutness" of the query, especially its focusing strength can be calibrated without introducing any semantic hierarchy from thesauri.

We observed the correlation between query length and effectiveness gained by supplemental phrasal indexing [Fujita 00a, Fujita 00b]. It is still in open question that such a difference of phrasal term effectiveness in different length of queries can be explained from the difference of "aboutness".

2.3 Reference Database as a Substitute for a Thesaurus

Since web queries are typically short and do not contain enough terms to discriminate documents, query expansion is desirable for the better results in TREC style evaluations.

For an automatic query expansion purpose, typically synonymous words from a thesaurus are utilized.

In Japanese text retrieval experiments, we once tried such a strategy and observed consistent but small improvement with a newspaper article database [Fujita 99b].

Such an approach is problematic since preparing and maintaining thesauri is not an easy task either for an open domain or a closed domain.

Another problem of utilizing pre-coded thesauri for query expansion is that synonymous relations described in thesauri are not necessarily mean equivalence as a query term. Semantic equivalence relations in lexicon level do not necessarily mean equivalence in subject concepts of retrieved documents.

Instead of such a semantic approach, documents themselves, which represent author's "aboutness" can be utilized as the source of query expansions. The technique is similar to pseudo-relevance feedback procedures, that is frequently used in TREC experiments but the database in pilot search is not identical to the retrieval target database itself. Since many web documents are terminologically so poor that it is natural to refer to other text sources for term extraction.

A reference database can be either general domain databases like newspaper or a specific domain database depending on the retrieval task in question.

In the case of web retrieval, a newspaper database seems to be appropriate, since it is open domain retrieval and the reference databases preferably cover the any subjects that might be in test topics. Only newspapers and encyclopaedia seem to possess such a broad coverage of content documents.

2.4 Another source of "aboutness": Anchor Text of Hyperlinks

When we ask what a page is talking about, sometimes anchor texts (or link texts, the texts on which a hyperlink is set) indicate exact and very short answer.

The anchor text is typically an explanation or denotation of the page that is linked to. Some commercial based search engines are utilizing such information for advanced searches [Altavista]. We treat anchor texts literally as the part of the linked document.

In total, 6,077,878 anchor texts are added to 1,173,189 linked pages out of 1,692,096 pages in the wt10g data set. So 69% document pages in the data set are attributed anchor text information on top of original page information.

3. SYSTEM DESCRIPTION

For the TREC-9 Web track experiments, we utilized the engine of Justsystem ConceptBase Search™ version 2.0 as the base system.

A dual Pentium III™ server (670MHz) running Windows NT™ server 4.0 with 1024MB memory and 136GB hard disk is used for experiments.

The document collections are indexed wholly automatically, and converted to inverted index files of terms.

3.1 Term Extraction

Queries and documents in target databases are analyzed by the same module that decomposes an input text stream into a word stream and parses it using simple linguistic rules , in order to compose possible noun phrases.

Extracted units are single word nouns as well as simple linguistic noun phrases that consist of a sequence of nouns or nouns preceded by adjectives.

3.2 Vector Space Retrieval

Each document is represented as a vector of weighted terms by tf*idf in inverted index files and the query is converted in similar ways.

Similarity between vectors representing a query and documents are computed using the dot-product measure, and documents are ranked according to decreasing order of RSV.

OKAPI BM25 function is utilized as TF part of weighting function [Robertson 94, Robertson 95] so that the retrieval process can be considered as probabilistic ranking.

3.3 Passage Retrieval

Since some pages are extremely long in the wt2g data set, we became aware of using passages rather than whole pages as the indexing unit is appropriate for the sake of retrieval effectiveness.

Passage delimiting is done by the manner that each passage becomes similar length rather than finding paragraph boundary.

3.4 Phrasal Indexing and Weighting

Our approach consists of utilizing noun phrases extracted by linguistic processing as supplementary indexing terms in addition to single word terms contained in phrases. Phrases and constituent single terms are treated in the same way, both as independent terms, where the frequency of each term is counted independently based on its occurrences .

As we indicated in [Fujita 99a, Fujita 00a], phrasal terms are over-weighted with normal scoring function. We evaluated the following three methods:

1) Empirical down-weighting method [Fujita 99a]

2) Fagan's method [Fagan 87]

3) Approximation to Robertson's method [Robertson 97]

As it performed always better than other methods in the pre-submission experiments, we adopted down-weighting approach although it requires empirical parameter tuning.

Another advantage of down-weighting approach is that the query specificity can be calibrated changing down-weighting parameters when enough phrasal terms are provided in the query.

3.5 Pseudo-Relevance Feedback and Reference Database Feedback

Automatic feedback strategy using pseudo-relevant documents is adopted for automatic query expansion.

The system submits the first query generated automatically from topic descriptions against the target or reference database, and considers the top n documents from relevant ranking list as relevant.

The term selection module extracts salient terms from these pseudo-relevant documents and adds them to the query vector.

Then the expanded query vector is submitted against the target database again and the final relevance ranking is obtained.

The whole retrieval procedure is as follows:

1) Automatic initial query construction from the topic description
2) 1st pilot search submitted against a reference database
3) Term extraction from pseudo-relevant documents and feedback
4) 2nd pilot search submitted against the target database
5) Term extraction from pseudo-relevant documents and feedback
6) Final search to obtain the final results

3.6 Term Selection

Each term in example documents are scored by some term frequency and document frequency based heuristics measures described in [Evans 93].

The terms thus scored are sorted in decreasing order of each score and cut off at a threshold determined empirically.

In effect, the following parameters in feedback procedures should be decided:

1) How many documents to be used for feedback?

2) Where to cut off ranked terms?

3) How to weight these additional terms?

These parameters are carefully adjusted using TREC-8 queries (topic 401-450), wt2g data set and their relevance judgement provided by NIST and 4 parameter sets for official runs are decided.

Run tag	Query	Link	Ref	Avg. Prec	R-Prec
jscbt9wcs1	VS	No	Yes	0.2011	0.2175
jscbt9wls1	VS	Yes	Yes	0.2000	0.2219
jscbt9wls2	VS	Yes	No	0.1838	0.2027
jscbt9wcl1	Long	No	Yes	0.2687	0.2841
jscbt9wll1	Long	Yes	Yes	0.2659	0.2812
jscbt9wll2	Long	Yes	No	0.2801	0.3054

Table 1: Performance of official runs

3.7 Spell Variation

Because of some spelling errors in "title" field texts of topic description, the system sometimes returned no document or few in very short query runs. In such a case, the initial queries are expanded automatically by generated spell variations.

The procedure consists of looking for similar words in the word lists extracted from the database. Spelling similarity is measured by a combination of uni-gram, bi-gram and tri-gram matching scores.

4. EXPERIMENTS

We submitted six automatic runs as follows:

jscbt9wcs1: Content only, very short query run with parameter set s1

jscbt9wls1: Link, very short query run with parameter set s1

jscbt9wls2: Link, long query run with parameter set s2

jscbt9wcl1: Content only, long query run with parameter set l1

jscbt9wll1: Link, long query run with parameter set l1

jscbt9wll2: Link, long query run with parameter set l2

As for the link run evaluation, we adopted "anchor text" of hyperlink information as some web search sites do.

The experiments are designed to measure effects of phrasal term indexing, pseudo-relevance feedback and reference database feedback with regards to different query types.

From our experience in NTCIR-1 experiments for Japanese text retrieval, we are paying attention to the relation between the effectiveness of elementary techniques and the query length.

We observed that performance gain by the pseudo-relevance feedback tend to be large when the query is shorter in NTCIR-1 experiments. It is easily understood that longer queries contain already so good terms that the feedback could no more find better terms in addition.

It seems more difficult to explain why supplemental phrasal indexing is more effective with longer queries.

4.1 Very Short Query Experiments

Very short query run using only "title" fields of topic description is recommended for all the sites.

The following settings are examined:

1. Content only, single words + phrases
2. Link, single words + phrases
3. Content only, single words
4. Link, single words

For each setting, combination of with/without reference database feedback and with/without pseudo-relevance feedback are examined with the same parameter set: s1, for the convenience of comparison. Results of 16 runs in total are compared in Table 2.

Since initial queries are very short (in average, 2.1 single word terms and 0.7 phrasal terms, maximum 5 single word terms and 3 phrasal terms , minimum 0 single word terms and 0 phrasal terms) and they do not contain enough terms, the automatic feedback procedure contributes to 4.5% to 7.5 % of consistent improvements in average precision in all cases.

The final queries contain 44.1 single word terms and 31.0 phrasal terms in average (maximum 138 single word terms and 176 phrasal terms, minimum 0 single word terms and 0 phrasal terms).

The improvement gained by the combination of a pseudo-relevance feedback and reference database feedback is 15.8% for content only run and 17.0% for link run.

Run description	Ref	PFB	AvgPrec	R-Prec
Content only / very short / SW + phrases	Yes	Yes	0.2028	0.2185
Content only / very short / SW + phrases	Yes	No	0.1893	0.2267
Content only / very short / SW + phrases	No	Yes	0.1849	0.2135
Content only / very short / SW + phrases	No	No	0.1751	0.2020
Link / very short / SW + phrases	Yes	Yes	0.2018	0.2228
Link / very short / SW + phrases	Yes	No	0.1927	0.2228
Link / very short / SW + phrases	No	Yes	0.1854	0.2082
Link / very short / SW + phrases	No	No	0.1725	0.1919
Content only / very short / Single words only	Yes	Yes	0.1864	0.1949
Content only / very short / Single words only	Yes	No	0.1714	0.1987
Content only / very short / Single words only	No	Yes	0.1763	0.2022
Content only / very short / Single words only	No	No	0.1683	0.2025
Link / very short / Single words only	Yes	Yes	0.1863	0.1976
Link / very short / Single words only	Yes	No	0.1732	0.1922
Link / very short / Single words only	No	Yes	0.1726	0.1948
Link / very short / Single words only	No	No	0.1693	0.1983

Table 2: Performance comparison (Very Short Query, s1 parameter set)

Supplemental phrasal indexing runs perform better in average precision both with/without pseudo-relevance feedback and with/without reference database feedback.

But without any feedback, single word runs are better in R-precision.

Again we confirmed the situation observed in Japanese text retrieval workshop NTCIR-1 [Fujita 99a], i.e. effectiveness of phrasal indexing is not clear when the queries are short.

Effectiveness of link run is not clear as well.

4.2 Long Query Experiments

Long query experiments examined queries automatically constructed from all fields in topic description.

Since TREC topic descriptions have a stratified explanation of topics in the sense that the subject explanations are iterated in different styles. Shorter fields contain only terms of "foreground aboutenss" and longer fields contain terms of "background aboutness" as well as terms of "foreground aboutness". It is important to adjust weighting for each term according to its "foregroundness" in the "request aboutness".

We adjusted term weights according to the fields in which the term appeared since this might be a good measure for term "foregroundness".

The same runs as very short query are examined:

1. Content only, single words + phrases
2. Link, single words + phrases
3. Content only, single words
4. Link , single words

The initial queries contain 11.6 single word terms and 3.46 phrasal terms in average (maximum 18 single word terms and 9 phrasal terms, minimum 5 single word terms and 0 phrasal terms) and the final queries contain 76.9 single word terms and 53.6 phrasal terms in average (maximum 239 single word terms and 218 phrasal terms, minimum 25 single word terms and 5 phrasal terms).

Table 3 shows the results. Supplemental phrasal runs are consistently better than single word term runs both in average precision and R-precision.

Since initial queries are longer and they contain terms of "background aboutness", performance improvements given by automatic feedback are comparatively smaller (o.3%-6.5%) than in very short query experiments (4.5%-7.5%).

No search effectiveness improvement by introducing feedback from a reference database is observed.

We reconfirmed our observation from Japanese text retrieval experiments that the phrasal term indexing is effective only with enough long initial topic description containing a certain number of phrases as well as single words, otherwise its effect is rather incidental.

Run description	Ref	PFB	AvgPrec	R-Prec
Content only / Long / SW + phrases	Yes	Yes	0.2666	0.2784
Content only / Long / SW + phrases	Yes	No	0.2612	0.2940
Content only / Long / SW + phrases	No	Yes	0.2771	0.3067
Content only / Long / SW + phrases	No	No	0.2649	0.3043
Link / Long / SW + phrases	Yes	Yes	0.2650	0.2861
Link / Long / SW + phrases	Yes	No	0.2642	0.2962
Link / Long / SW + phrases	No	Yes	0.2801	0.3054
Link / Long / SW + phrases	No	No	0.2631	0.2942
Content only / Long / Single words only	Yes	Yes	0.2486	0.2518
Content only / Long / Single words only	Yes	No	0.2516	0.2793
Content only / Long / Single words only	No	Yes	0.2568	0.2883
Content only / Long / Single words only	No	No	0.2456	0.2762
Link / Long / Single words only	Yes	Yes	0.2480	0.2538
Link / Long / Single words only	Yes	No	0.2534	0.2772
Link / Long / Single words only	No	Yes	0.2614	0.2882
Link / Long / Single words only	No	No	0.2449	0.2729

Table 3: Performance comparison (Long query, 12 parameter set)

As in the very short query runs, it is not clear at all if link runs are better or not than content only runs. In the pre-submission experiments with the wt2g database and TREC-8 topics, small but consistent improvement was observed, but it is not the case with the TREC-9 main web test set. We did not yet find enough reason for this.

5. CONCLUSIONS

TREC-9 experiments at Justsystem group are described.

The following conclusions are drawn from these experiments:

1) Phrasal indexing seems to be more effective when the query is longer.

2) Pseudo-relevance feedback always contributes to the performance especially when initial queries are very short.

3) Feedback from a reference database was effective with very short queries but not with long queries.

4) No reliable performance improvement utilizing anchor texts was observed in wt10g experiments. Sometimes it was effective but not always.

On the other hand, we need more experiments as well as careful observation on the effect of phrasal indexing with short queries.

It is also interesting to compare the effects of reference database feedback with query expansion by WordNet style pre-coded thesauri.

For the future work, it is desirable to introduce the distinction of foreground/background of "aboutness" in question answering task where identification of focus of the topic description is crucial.

6. ACKNOWLEDGMENTS

Our thanks to Mr. Toshiya Ueda and Mr. Tatsuo Kato for their assistance.

REFERENCES

[1] Altavista: http://doc.altavista.com/adv_search/ast_ma_clickhere.html

[2] Evans, D.A. and Lefferts, R.G., Grefenstette, G., Handerson, S.K., Hersh, W.R., and Archbold, A.A., CLARIT TREC Design, Experiments and Results, in Proceedings of the First Text REtrieval Conference(TREC-1), NIST Special Publication 500-207, Washington D.C., 1993,,494-501.

[3] Fagan, J.L. Experiments in Automatic Phrase Indexing for Document Retrieval: A Comparison of Syntactic and Non-syntactic Methods, Ph.D Thesis,

Dept. of Computer Science, Cornell University, Sept. 1987.

[4] Fujita, S. Notes on Phrasal Indexing—JSCB Evaluation Experiments at NTCIR AD HOC, in Proceedings of NTCIR-1 workshop, 1999.

[5] Fujita, S. Notes on Phrasal Indexing: JSCB Evaluation Experiments at IREX-IR, in Proceedings of IREX Workshop, Tokyo, 1999, 45-51.

[6] Fujita, S. Evaluation of Japanese Phrasal Indexing with a Large Test Collection, in RIAO2000 Conference proceedings, Paris, 2000, 1089-1098.

[7] Fujita, S. Discriminative Power and Retrieval Effectiveness of Phrasal Indexing Terms, in Proceedings of the ACL-2000 Workshop on Recent Advances in Natural Language Processing and Information Retrieval, Hong Kong, 2000, 47-55.

[8] Ingwersen, P. Information Retrieval Interaction, Taylor Graham Publishing, London, 1993.

[9] Lewis, D. Representation and Learning in Information Retrieval, Ph.D Thesis, Dept. of Computer and Information Science, University of Massachusetts, Feb. 1992.

[10] Lewis, D. An Evaluation of Phrasal and Clustered representation on a Text Categorization Task, in Proceedings of the Fifteenth Annual International ACM SIGIR Conference(Copenhagen, June 1992), ACM Press, 37-50.

[11] Robertson, S.E. and Walker S. Some Simple Effective Approximations to the 2Poisson Model for Probabilistic Weighted Retrieval, in Proceedings of the Seventeenth Annual International ACM SIGIR Conference(Dublin, July 1994), Springer-Verlag, 232-241.

[12] Robertson, S.E., Walker S., Jones S., Hancock-Beaulieu, M.M., Gatford, M. Okapi at TREC-3, in Proceedings of the Third Text REtrieval Conference(TREC-3), NIST Special Publication 500-225, Washington D.C., 1995, 109-126.

[13] Robertson, S.E. and Walker S. On relevance weights with little relevance information, in Proceedings of the 20th Annual International ACM SIGIR Conference(Philadelphia, July 1997), ACM Press, 16-24.

Appendix A.

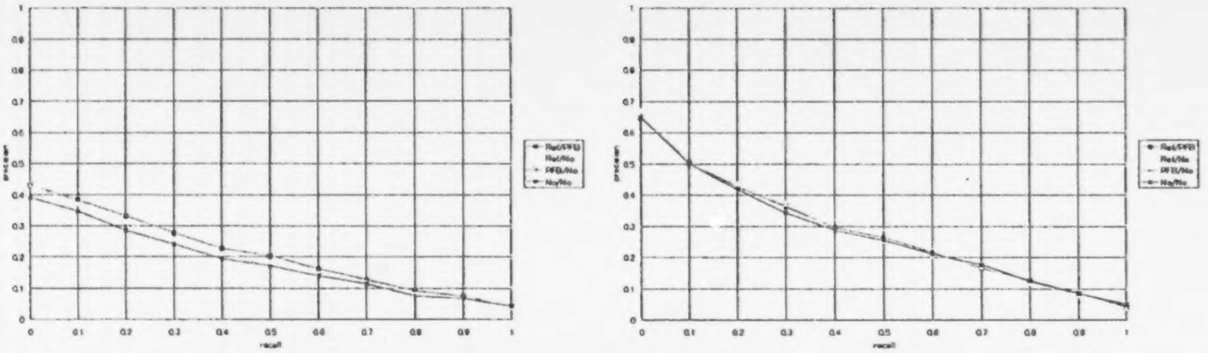


Figure 1: Recall-precision curves of supplemental phrasal runs
Left: Content only very short runs, Right: Link long runs

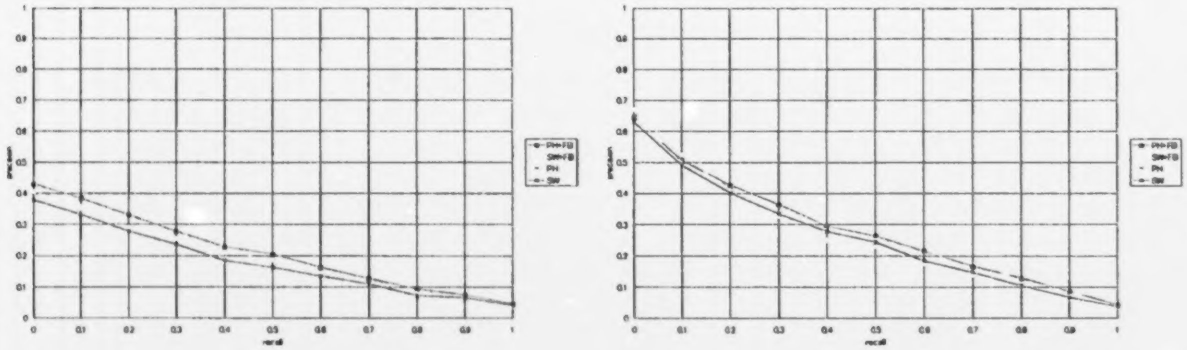


Figure 2: Recall-precision curves of supplemental phrasal runs vs single word runs with/without feedback Left: Content only very short runs Right: Link long runs

Experiments on the TREC-9 Filtering Track

Keiichiro Hoashi† Kazunori Matsumoto† Naomi Inoue† Kazuo Hashimoto†
Takashi Hasegawa‡ Katsuhiko Shirai‡

KDD R&D Laboratories, Inc.†
2-1-15 Ohara Kamifukuoka, Saitama 356-8502, Japan

School of Science and Engineering, Waseda University‡
3-4-1 Okubo Shinjuku, Tokyo 169-8555, Japan

1 Introduction

KDD R&D Laboratories has been participating in previous TREC conferences with the cooperation of students from Waseda University. This year, KDD R&D Laboratories and Waseda University are officially participating as a joint research team.

We have focused our experiments for TREC-9 on the adaptive filtering experiments of the Filtering Track. Our goal was to evaluate the filtering method using a non-relevant information profile. We have also made experiments of a new feedback method to increase the accuracy of pseudo feedback. In this paper, we will describe our filtering methods, and present results of our evaluations.

2 Filtering methods

In this section, we will describe the filtering methods used in our experiments, and present some results from previous TREC experiments for background.

2.1 Profile updating using word contribution

Query expansion method using word contribution was applied to the profile updating process of our filtering system. Word contribution (WC) is a measure to express the influence of a word to query-document similarity. WC is defined by the following formula:

$$Cont(w,q,d) = Sim(q,d) - Sim(q'(w), d'(w)) \tag{1}$$

where $Cont(w,q,d)$ is the contribution of the word w in the similarity between query q and document d, $Sim(q,d)$ is the similarity between q and d, $q'(w)$ is query q excluding word w, and $d'(w)$ is document d excluding word w. In other words, the contribution of word w is the difference between the similarity of q and d, and the similarity of q and d when word w is assumed to be nonexistent in both data. Therefore, there are words which have positive contribution, and words which have negative contribution. Words with positive contribution raise similarity, and words with negative contribution lower similarity.

Analysis on WC[3] show that words with either highly positive or negative contribution are few, and that most words have contribution near zero. This means that most words do not have a significant influence on query-document similarity. As obvious from the definition of word contribution, words with highly positive contribution are words which cooccur in the query and document. Such words can be considered as informative words of document relevance to the query. On the contrary, words with highly negative contribution can be considered as words which discriminate relevant documents from other non-relevant documents contained in the data collection.

In the query expansion method based on WC, words used for QE were extracted only from relevant documents. In the profile updating method based on WC[1], information from all selected documents were used, regardless of their relevance to the profile.

First, the word contribution of all words in the selected document are calculated. From each selected document d, N words with the lowest contribution are extracted. Next, a score for each extracted word w is calculated by the following formula:

$$Score(w) = wgt \times Cont(w, p, d) \tag{2}$$

where wgt is a parameter with a negative value (since the contribution of the extracted word is also negative), and $Cont(w, p, d)$ is the WC of word w to the similarity of profile p and document d. On this procedure, the calculated score is regarded as the TF (term frequency) element of the word. Finally, all extracted words and their weights are added to the profile, unless the calculated weight of the word is negative.

A Rocchio-like algorithm[6] is applied here to add information from non-relevant documents to the profile. When the selected document d is relevant to the profile, the weight of word w is added to the element of the profile vector which expresses w. When d is non-relevant, the weight is subtracted from the element of the profile vector. Seperate parameters (wgt) are used for the calculation of $Score(w)$ described in Formula (2), depending on the relevance of d. wgt_{relR} is the parameter for words extracted from relevant documents, and wgt_{nrelR} is the parameter for words extracted from non-relevant documents.

Elements of the profile vector with negative weights are not used for similarity calculation, but all weights are accumulated for profile updating on upcoming documents. Therefore, the weights of words which appear in both relevant and non-relevant documents are restrained, thus emphasizing words which only appear in relevant documents.

2.2 Filtering method using non-relevant information profile

To improve filtering performance without sacrificing retrieval of relevant documents, it is necessary to reduce non-relevant document selection. However, the analysis on results of the experiments described in the previous section showed that this is difficult when filtering is based on only the similarity between the profile and incoming documents, as in most existing filtering systems.

In order to reduce retrieval of non-relevant documents, we have proposed the use of a profile which expresses the features of non-relevant documents[4]. By calculating the similarity between this ***non-relevant information profile*** and incoming documents which have passed the initial profile, and rejecting documents which have high similarity to the non-relevant information profile, it is possible to avoid selection of documents highly similar to past retrieved non-relevant documents. By rejecting such documents, improvement of filtering performance is expected.

The process flow of filtering with the non-relevant information profile is illustrated in Figure 1, where d is the selected document, p_R is the initial profile, p_N is the non-relevant information profile, and $Sim(p, d)$ is the similarity between profile p and document d.

As illustrated in Figure 1, thresholds $Thres_R$ and $Thres_N$ are set for each profile. The similarity between p_N and documents which have passed p_R is calculated, and compared to

BEST COPY AVAILABLE

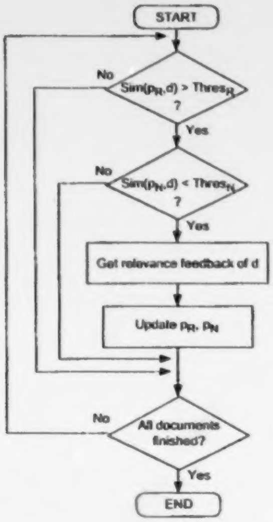


Figure 1: Filtering process with non-relevant information profile

$Thres_N$. If the similarity exceeds $Thres_N$, then the document is regarded as non-relevant, and, as a result, is rejected by p_N.

The method to build the non-relevant information profile is as the following:

Initial values of all elements in the non-relevant information profile are set to 0. For each selected document, N words are extracted and their weights are calculated based on WC. As in the original WC-based profile updating method, parameter wgt differs based on the relevance of the selected document. For the generation and updating of p_N, wgt_{relN} is the parameter for words extracted from relevant documents, and wgt_{nrelN} is the parameter for words extracted from non-relevant documents. To update the non-relevant information profile, the weights of words extracted from non-relevant documents are added, and weights of words extracted from relevant documents are subtracted from the regarding element of the profile vector. This is opposite from the updating of the initial profile, where the weights of words extracted from relevant documents were added to the regarding element of the profile vector, and the weights of words extracted from non-relevant documents were subtracted.

In addition to the updating of the non-relevant information profile, the initial profile p_R is also updated by the method described in Section 2.1.

2.3 Updating non-relevant profile with pseudo feedback

Results from the experiments described in the previous section show that there is a tradeoff between the strictness of $Thres_N$ and the performance of profile p_N. If $Thres_N$ is set at a low value, the number of documents blocked by p_N. This leads to the decrease of feedback information to the profile, which correlates to the performance of the filter itself. However, if $Thres_N$ is raised to increase feedback information, the number of documents rejected by p_N will also decrease, thus making the increase of feedback meaningless. To solve this problem, we

BEST COPY AVAILABLE

propose the use of pseudo feedback[5] to increase feedback information.

Pseudo feedback is often used for QE in the text retrieval task, when the relevance of retrieved documents is uncertain. Generally, documents which are high-ranked on the initial search are assumed to be relevant. This assumation is sent back to the system, which utilizes this information to expand the query.

Our proposal is to assume documents that are blocked by p_N as non-relevant, and to send this information to the profile updating process. The documents regarded as non-relevant by pseudo feedback are handled as the same as documents which were actually regarded non-relevant from the original relevance feedback. This method allows $Thres_N$ to be strict without sacrificing feedback information.

2.4 Weighting pseudo feedback information

Our experiments with TREC-8 filtering data proved that pseudo feedback was effective. However, the number of relevant documents mistakenly rejected by the filtering system had increased by the implementation of pseudo feedback. This is caused by the inaccuracy of pseudo feedback information. Some relevant documents were mistakenly regarded as non-relevant in the pseudo feedback process, leading to mistaken feedback information to the profile.

In order to solve this problem, we propose the weighting of pseudo feedback information. Documents with high similarity to the non-relevant information profile have a higher probability to be actually non-relevant, compared to documents with low similarity to the non-relevant information profile. Our method applies a weight to the documents which pseudo feedback occurs from, based on the similarity between the document and the non-relevant information profile.

The weighting method is expressed by the following formula:

$$Value_{new}(w_i) = Value_{org}(w_i) \times \frac{sim_N - Thres_N}{1 - Thres_N} \qquad (3)$$

where $Value_{org}(w_i)$ expresses the original feedback value for word w_i extracted by previously described methods, and $Value_{new}(w_i)$ expresses the feedback value weighted by our proposed method. In this formula, we multiply a weight to the originally extracted value. The weight is a normalized value of Sim_N, i.e., the similarity between the document and the non-relevant information profile. This method emphasizes pseudo feedback information extracted from documents which are assumed to have a high probability to be non-relevant to the profile, and reduce feedback information from "suspective" documents. Therefore, the improvement of the quality of pseudo feedback can be expected.

2.5 Additional System Details

Our system is based on the vector space model. The weighting calculation scheme is based on the TF∗IDF based weighting formulas for the SMART system at TREC-7 [7], with minor customizations. The TF and IDF factors for our system are as the following:

- TF factor

$$\log(1 + tf) \qquad (4)$$

- IDF factor

$$\log\left(\frac{M}{df}\right) \qquad (5)$$

where tf is the term's frequency in the document, df is the number of documents that contain the term, and M is the total number of documents in the data collection. The document frequency data was generated from TREC CD-ROMs Vol 4 and 5. We have added 1 to the term frequency inside the logarithm of the TF factor because the tf value resulting from word contribution occasionally has values below 1, which results in a negative weight.

3 Experiments

3.1 Conditions

As previously mentioned, we have focused our experiments on the adaptive task. Furthermore, we have only made experiments with the OHSUMED topic set.

Parameters for updating the initial profile (wgt_{relR}, wgt_{nrelR}) were fixed to -800 and -200, respectively. These values were derived from preliminary experiments on the original single-filter algorithm described in Section 2.1.

Parameters for the non-relevant information profile were set as the following: $wgt_{relN} = \{-200, -400, -800\}$, $wgt_{nrelN} = \{-100, -200, -400, -800\}$. The threshold for the initial profile $Thres_R$ was set at 0.1, and the threshold for the non-relevant information profile was set at 0.25. The thresholds were set at a moderate value in order to increase the retrieval of documents, so there will be sufficient data for analysis of the filtering process.

Using the parameters listed above, we ran experiments for the normal filtering method using the non-relevant information profile (*Normal*), the pseudo feedback method (*Pseudo*), and the method with weighting applied to pseudo feedback (*Weight*).

3.2 Results

Tables 1 to 3 show the average scaled utility (T9U) of the *Normal*, *Pseudo*, and *Weight* methods for each set of wgt_{nrel} parameters. The results officially submitted to TREC are written in bold font.

Table 1: Average scaled utility (T9U), *Normal*

	wgt_{nrelN}			
w_{nrelR}	-100	-200	-400	-800
-200	0.5570	0.5584	0.5637	**0.5662**
-400	0.5569	0.5574	0.5606	**0.5631**
-800	0.5551	0.5578	0.5591	0.5612
1-filter	0.5126			

The results in Tables 1 to 3 show that the non-relevant information profile was effective in improving filtering performance. However, we could not observe significant difference between the 3 methods using the non-relevant information profile, although the pseudo feedback weighting method had the best overall scaled utility.

Moreover, it can be observed that all methods with use of the non-relevant information profile achieved higher performance when the wgt_{nrelN} parameter was set at a higher absolute value than wgt_{nrelR}. This shows that the performance of the non-relevant information profile is better when feedback information from non-relevant documents are emphasized.

Table 2: Average scaled utility (T9U), *Pseudo*

	wgt_{nrelN}			
w_{nrelR}	-100	-200	-400	-800
-200	0.5567	0.5593	0.5622	**0.5655**
-400	0.5568	0.5585	0.5597	**0.5645**
-800	0.5560	0.5601	0.5593	0.5617
I-filter	0.5126			

Table 3: Average scaled utility (T9U), *Weight*

	wgt_{nrelN}			
w_{nrelR}	-100	-200	-400	-800
-200	0.5576	0.5576	0.5652	**0.5661**
-400	0.5578	0.5588	0.5623	**0.5650**
-800	0.5560	0.5594	0.5599	0.5635
I-filter	0.5126			

4 Discussion

Results from our experiments are somewhat similar to the results observed from our TREC-8 Filtering experiments, in which the system achieved higher (utility-wise) performance as the threshold became more strict. Therefore, we were refrained from exploring new research themes such as dynamic threshold adjustment, because the threshold will automatically converge to an extreme level if the threshold adjustment method was planned to be optimized based on utility. However, dynamic threshold adjustment is an obviously effective technique for achieving high filtering performance. We believe we have proved the effectiveness of the non-relevant information profile through our experiments, so our next step will be to implement threshold adjustment to our filtering system.

References

[1] K Hoashi, K Matsumoto, N Inoue, K Hashimoto: "Experiments on the TREC-8 Filtering Track", (to be published in *The 8th Text REtrieval Conference*"), 2000.

[2] K Hoashi, K Matsumoto, N Inoue, K Hashimoto: "TREC-7 Experiments: Query Expansion Method Based on Word Contribution", The 7th Text REtrieval Conference, NIST SP 500-242, pp 433-441, 1999.

[3] K Hoashi, K Matsumoto, N Inoue, K Hashimoto: "Query Expansion Method Based on Word Contribution", Proceedings of SIGIR'99, pp 303-304, 1999.

[4] K Hoashi, K Matsumoto, N Inoue, K Hashimoto: "Document Filtering Method Using Non-Relevant Information Profile", Proceedings of ACM-SIGIR 2000, pp 176-183, 2000.

[5] S Robertson, S Walker, S Jones, M Hancock-Beaulieu, and M Gatford, "Okapi at TREC-3", Overview of the Third Text REtrieval Conference, pp 109-125, 1994.

[6] J Rocchio: "Relevance Feedback in Information Retrieval", in "The SMART Retrieval System – Experiments in Automatic Document Processing", Prentice Hall Inc., pp 313-323, 1971.

[7] A Singhal, J Choi, D Hindle, D Lewis, and F Pereira: "AT&T at TREC-7", The Seventh Text REtrieval Conference, pp 239-251, 1999.

TREC-9 Experiments at KAIST: QA, CLIR and Batch Filtering

Kyung-Soon Lee, Jong-Hoon Oh, JinXia Huang, Jae-Ho Kim and Key-Sun Choi

Division of Computer Science,
Department of Electrical Engineering & Computer Science,
Korea Advanced Institute of Science and Technology, KORTERM,
373-1 Kusung Yusong Taejon 305-701 Korea
Tel: +82-42-869-5565
Fax: +82-42-867-3565
{kslee, rovellia, hgh, jjaeh, kschoi}@world.kaist.ac.kr

1. Introduction

In TREC-9, we participated in three tasks: question answering task, cross-language retrieval task, and batch filtering task in the filtering task.

Our question answering system consists of following basic components - query analyzer, Named entity tagger, Answer Extractor. First, question analyzer analyzes the given question. Question analyzer generates question type and keywords of the given question. Then retrieved documents are analyzed for extracting relevant answer. POS tagger and Named entity tagger are used for the purpose. Finally, Answer Extractor generates relevant answer.

There are four runs in our CLIR, two runs follow the dictionary and MI information based translation approach (KAIST9xlqm, KAIST9xlqt), another one using the mixture result of two commercial Machine Translation systems (KAIST9xlmt), and the final one is monolingual run (KAIST9xlch). We translated only query and description fields in all four runs.

In batching filtering task, we submitted results for OHSU topics and MSH-SMP topics. For OHSU topics, we have been exploring a filtering technique which combines query zone, support vector machine, and Rocchio's algorithm. For MSH-SMP topics, we use support vector machine simply.

2. Question Answering track

2.1 System Description

Our TREC-9 question answering system consists of three basic components - query analyzer, named entity tagger, and answer extractor. Our system operates on a set of documents retrieved by information retrieval system. For convenience, we worked with the top-ranked document set generated by NIST.

First, a question analyzer analyzes the given question. It generates question type and extracts keywords of the given question. Then top 50 documents retrieved by information retrieval system are analyzed for extracting relevant answer. A POS tagger and a named entity tagger are used for the purpose. Finally, an answer extractor generates relevant answers from named entity tagged documents using question types and keywords analyzed by the question analyzer.

2.1.1 Question Analyzer

A question analyzer parses the given question to identify question types and extract keywords. We define six kinds of question type for the expected answer.

<Person>, <Location>, <Organization>, <Time>, <Currency>, <Measure>

There are five steps for analyzing questions. First, a question is tagged by POS tagger – We use the Brill tagger (Brill, 1995). Second, keywords are extracted from tagged question. The POS tag, which we extract as keywords, is noun, adjective, countable numeric, and verb. However, we exclude category of "be-verb (is, are, was, were)" and "do-verb (do, does, did)". Third, we check an acronym in the given question. If there is a word with capital letters, we assume that it is an acronym and we search an expanded form in the acronym dictionary. Once, there is an expanded form, we add it to keyword lists. For example, "Cable News Network", which is expanded form of CNN, is added into keyword list for the question containing word 'CNN'. Fourth, a question type is determined by the pattern that we define. There are list of patterns in the table 2.1.

Table 2.1 Patterns for each question type

Question Type	Patterns
Person	Who, Who's, Whom~, ~man's name, ~woman's name
Location	Where ~, What + location (city, country,.....) ~ In what + location (city, country....) ~, What nationality~
Organization	What company~, What institution~
Time	When~, What time~, How many + Time (years, months, days)~
Currency	How much ~ spend, rent, cost, money, price~
Measure	How much, How many, How+ Adjective

If there is no matched pattern in the question, we estimate its question type using WordNet (Miller *et al.* 1991). We extract a noun phrase, which contains a head noun of the given question and estimate its question type based on synsets and a hypernyms of the head noun. Table 2.2 shows synsets and hypernyms lists for each category. If the synsets and hyperyms of the head noun are not matched with a list in the table 2.2, we generate a noun phrase, which contains a head noun of question, as question type.

Table 2.2 Lists of WordNet synsets and hypernyms to assign questions to question types

Question type	Synsets and hypernyms
Person	Person
Location	district, territory, region
Organization	Commercial
Time	time period
Currency	cost, price
Measure	measure, magnitude

2.1.2 Named Entity Tagger

For the given POS tagged text, a named entity tagger generates named-entity tagged texts. It identifies six kinds of question type, which we define in the question analyzer step: Person = <PER>, location, region = <LOC>, organization = <ORG>, date or time expression = <TIME>, expression containing currency = <CUR>, and measures expression = <MEA>. For detecting question type of <PER>, <LOC>, and <ORG> - which are proper nouns, we use dictionaries for them. There are about 50,000 entries for person, 1,300 entries for location and 4,000 entries for organization. If we can not determine the type of a named entity, we tag it as <NPP>. And we use patterns and dictionaries for identifying question type of <TIME>, <CUR>, and <MEA>. For applying patterns, we extract phrase using regular expression - DT JJ* CD*. In the regular expression, DT, JJ and CD represent determiner, adjective, and cardinal number respectively. Table 2.3 shows patterns for <TIME>, <CUR>, and <MEA>.

Table 2.3 Patterns for "Time", "Currency", and "Measure" Named Entity

Question type (Named Entity)	Pattern
<TIME>	Four sequential digit e.g. 1942, Four sequential digit + 's' Four sequential digit + punctuation mark, Four sequential digit + 's' + punctuation mark mid- or late- , in + sequential
<CUR>	$+digit
<MEA>	digit +(m, km, cm), digit +(kg,g), digit+ l(liter)

2.1.3 Answer Extractor

Our answer extractor generates top-5 ranked phrases with two steps. First we extract three sentences for each document using keywords and question types. Second, sentences are partitioned into fixed length phrases – under 50 bytes and under 250 bytes. Third, the partitioned phrases are scored by keywords and question types.

Sentence Selection

In sentence selection step, top-3 sentences are selected for each document. Since, we believe that the context is very important for selecting relevant sentence, we consider the previous sentence, the current sentence and the next sentence for selecting relevant sentence. Each sentence is scored using a keyword and a question type of the given question. We use W_{sent} in the formula (3-1) for scoring sentences. It is based on the fact that how many keywords are matched and how many question types (named entity) are relevant to the question type of the given question in the current sentence, the previous sentence, and the next sentence. We believe that more keywords appear in a sentence and a context of the sentence and there are more relevant question types (named entities) to the question, the sentence has higher probability to contain a relevant answer to the question.

$$W_{key}(S_{Qij}) = \frac{matched \quad keywords \quad in \quad S_{Qij}}{\#of \quad keywords \quad in \quad Q_i}$$

$$W_{NE}(S_{Qij}) = \frac{\#of \quad NE \quad matched \quad with \quad Qtype_{Qi}}{\#of \quad NE \quad in \quad S_{Qij}}$$

$$W_{context}(S_{Q_{ij}}) = (W_{key}(S_{Qij}) + 0.5 \times W_{NE} \times (S_{Qij}))$$

$$W_{Sent}(S_{Qij}) = \alpha \times W_{context}(S_{Q_{ij-1}}) + \beta \times W_{context}(S_{Q_{ij}}) + \gamma \times W_{context}(S_{Q_{ij+1}}) \qquad (3-1)$$

where, S_{Qij} is jth sentence for the question i, and Qi is ith question.

Phrase Selection

In this step, we partition the extracted sentences into phrases with fixed length (under 50 bytes and under 250 bytes). Then each phrase is scored using keywords and question types. And top-5 ranked phrases are extracted as the relevant answer to the given question. We score phrases using W_{pass} in the formula (3-2). In the formula, we use nine window contexts for calculating scores of the phrases. It means that the contexts of the current phrase are considered – the previous four phrases and the next four phrases.

$$W_{key}(P_{Qij}) = \frac{matched \quad keywords \quad in \quad P_{Qij}}{\#of \quad keywords \quad in \quad Q_i}$$

$$W_{NE}(P_{Qij}) = \frac{\#of \quad NE \quad matched \quad with \quad Qtype_{Qi}}{\#of \quad NE \quad in \quad P_{Qij}}$$

$$W_{context}(P_{Qij}) = W_{key}(P_{Qij}) + 0.5 \times W_{NE}(P_{Qij})$$

$$W_{Pass}(P_{Qij}) = \theta_1 \times \sum_{P=P_{Qij-4}}^{P_{Qij-1}} W_{context}(P) + \theta_2 \times Wcontext(P_{Qij}) + \theta_3 \times \sum_{P=P_{Qij+1}}^{P_{Qij+4}} W_{context}(P) \qquad (3-2)$$

where, P_{Qij} is jth phrase for the question i, and Qi is ith question.

2.2 Results

2.2.1 Performance of Our QA system

We submitted one run in the 50byte category and one run in the 250byte category. The results are presented in Table 2.4. Table 2.4 breaks down the results by question type. Our system answers more correctly for the "Organization", "Currency", and "Person" question type than others. Not surprisingly, most of question types produce better result for the 250byte run than the 50byte run. However, for the question types, "Measure", "Time", and "Currency", there is not significant performance increase in the 250 bytes run result. We believe that the named entity tagger of our system can not identify the question types – "Measure", "Time", and "Currency" - very well and it makes difficult for answer extractor to identify relevant answers in the texts.

Table 2.4 Performance of our system for each question type. ARR means "Average Reciprocal Rank"

Question Type	# of Question	50 bytes run type			250 bytes run type		
		Correct #	Correct %	ARR	Correct #	Correct %	ARR
Person	147	52	35.37%	0.2355	**80**	**52.98%**	**0.3496**
Location	137	*41*	*29.93%*	*0.1915*	59	43.07%	0.3018
Organization	14	**7**	**50%**	**0.4071**	**8**	**57.14%**	**0.4524**
Time	77	25	32.47%	0.2353	*31*	*39.24%*	*0.2753*
Currency	6	**5**	**83.33%**	**0.4861**	**4**	**66.7%**	**0.45**
Measure	62	22	35.48%	0.2788	*19*	*30.16%*	*0.2193*
Other	239	*62*	*25.94%*	*0.1651*	**119**	**48.97%**	**0.3466**
Total	682	214	31.4%	0.212	320	46.9%	0.327

2.2.2 Error Analysis

We perform error analysis on the first 100 questions. It focuses on 250bytes run results. We divide errors into four types according to the component of our system where it causes errors. Table 2.5 shows errors and error types in the 250 bytes run results on the first 100 questions.

Table 2.5. Error analysis on the first 100 questions.

Error Type	# of Error (% of Error)
IR (Information retrieval) error	17 (34%)
QA (Query Analyzer) error	5 (10%)
NE (Named Entity Tagging) error	6 (12%)
AE (Answer Extraction) error	22 (44%)
Total	50

The first one is an IR type error. If there are no relevant answers in the retrieved documents, we determine the error as the IR error. For example, for "*Q222: What is Anubis?*", there is no relevant answer in the retrieved document. There are many errors with IR error type. We believe that since, the retrieved documents are very small, – they are only 50 documents for each question –, and there are many question, which are very short, – there is only one content word such as "*Q236: Who is Coronado?*", and "*Q241: What is a caldera?*" –, IR system can not retrieve relevant documents very well.

The second one is a QA type error. If query analyzer mis-analyses the given question, we call it a QA error. For example, "*Q288: How fast can a Corvette go?*" is analyzed as "Other" by the question analyzer, although it should be "Measure" question type. It is caused by a POS tagger error – "fast" is tagged as a noun. Therefore, the question analyzer produces a wrong result, although there is the pattern, [How + adjective => "Measure" question type].

The third one is a NE type error. We treat errors as the NE error when named entity tagger can not detect the relevant named entity to the question type in the sentence or phrase where answer appears. We exclude the case that named entity tagger mis-analyses the named-entity boundaries or can not identify the precise named entity in the sentence or phrase where answer does not appear. It is because there are too many named entities to check them.

For example, for "*Q209: Who invented the paper clip?*", the relevant answer is located in the following sentence.

> The paper clip, weighing <CUR>a desk-crushing 1,320 pounds,</CUR> is a faithful copy of **<NPP>Norwegian Johan Vaaler's</NPP>** <TIME>1899</TIME> invention, said Per <NPP>Langaker</NPP> of <NPP>the Norwegian School</NPP> of <NPP>Management.</NPP>

The question is analyzed as the "Person" question type. Therefore, answer will be "Person" named entity. However, "Norwegian Johan Vaaler", which can be relevant answer, is tagged as "<NPP>"- it means that the type of the named entity can not be determined.

There is another kind of NE type error. It is caused by roughly categorized question type.

For example, for "*Q245: Where can you find the Venus flytrap?*", the question can be treated as "Location" question type. And following sentence can be its answer.

> "**Whole savannas where flytraps were abundant** have been cleaned out," says <PER>Cecil Frost,</PER> coordinator of <PER>the North Carolina Plant Conservation Program.</PER>

Since, named entity tagger identifies "Location" named entities when it tagged as proper noun, the words, which contains meaning of location and is not proper noun, can not be detected as a "Location" named entity. Therefore, "savannas" is not tagged as the "Location" named entity and we can not extract it as answer.

The fourth one is an AE type error. When answer extractor can not identify the relevant answers, we define it as the AE error. Since, our answer extractor system extracts three sentences for each document, we can not extract the answer, which appear in the multi-sentence in the document. For example, for the question "*Q203: How much folic acid should an expectant mother get daily?*", following sentences can be relevant answer.

> Here are some good sources of folic acid according to <NPP>the USDA.</NPP>
> Raw forms of some vegetables are not included, because in their raw state they don't contain enough folic acid. For example, a 1/2-cup cooked serving of beets contains more than the same amount of the vegetable raw. <NPP>Also,</NPP> the term "good source" is based on **<NPP>the RDA</NPP> of <MEA>400</MEA> micrograms daily for a pregnant woman.**

In the sentence, which contains relevant answer, there is no words that matched with keyword analyzed by the question analyzer – *folic, acid, expectant, mother, get*. However, through four sentences, there are words matched with the keywords.

2.3. Discussion

Since our team did not have experience in the development of Question-Answering system before participating in the QA-track this year, our system is open to further improvement. Among the research issue for improving performance of our system, we will focus on following aspects:

- More detailed question type – we will divide each question type into detailed question type.
- Different weighting schemes for extracting sentence and phrase in the documents.
- Sorting criteria for the equally scored phrase and sentence.
- Sophisticated named entity tagger – using machine-learning technique.
- Coreference resolution
- Multi-sentential answer extraction

3. Cross-Language Information Retrieval track

There are four runs in our CLIR, two runs follow the dictionary and MI information based translation approach (KAIST9xlqm, KAIST9xlqt), another one using the mixture result of two commercial Machine Translation systems (KAIST9xlmt), and the final one is monolingual run (KAIST9xlch). We translated only query and description fields in all four runs.

We used SMART system (Salton, 1983) in our IR part after query translation. And because most of our resources are in GB code, we converted all BIG5 documents and the Chinese topics that given by TREC9 to GB. We used Universal Code Converter of shareware NJStar Communicator 2.0 (http://www.njstar.com/).

3.1 Translation Approach

3.1.1 Dictionary and MI information based query translation

The first two cross-lingual runs - KAIST9xlqm and KAIST9xlqt follow next steps:

1. Preprocessing.

   Paring the English topics and descriptions using the parser of Brill tagger (Brill, 1995), remain only noun and noun phrases.

2. Translation.

   Translate the remaining noun and noun phrases to Chinese using dictionary, do segmentation after translation. For example of CH55, we got "Word Trade Organization/f/世界贸易组织","membership/n/会员资格，成员资格" from dictionary, after segmentation, the probable translations of "Word Trade Organization member ship" will be "Word Trade Organization membership : 世界 贸易 组织 会员 资格, 世界 贸易 组织 成员 资格".

Proper noun translation was quite a problem in TREC9 query. In our translation, the proper noun recognition and translation followed next steps:

1) If a capitalized word cannot be found in our bilingual dictionary, and it satisfies a Chinese Pinyin spelling, it will be regard as a Chinese proper noun (ex, Wan). If a Chinese proper noun is a Pinyin sequence contains more than two characters, it will be separated (ex, Daya → Da Ya), but after translation, them will be considered as one Chinese proper noun again. If a Chinese proper noun is followed by another Chinese proper noun, regard them as one word after translating (ex, "Da Ya"+"Wan" → "Da Ya Wan").

2) Getting all probable Chinese characters of the Pinyin sequence by using Chinese Pinyin–character table, and select the Chinese character associations by using the character co-occurrence information that can be gotten from Chinese corpus. Chinese dictionary will be used in this step to delete the common words from the probable Chinese character associations.

   If the Chinese Proper noun contains only two characters, it will be a one-stop process: get all probable associations, delete the associations that can be found in Chinese dictionary – because they will be common words, and then get the occurrence times of the associations from Chinese corpus, remain the association that has the most frequent occurrence.

   If it contains more than three characters, we will get the occurrence time of first two character's first, delete common words from them, and remain only the associations that the occurrence times are ranking in top 5%. Then combine the remain associations to third probable characters, delete common words again, remain top 5% associations, and so on. In the final step, only the character association that owns the highest occurrence time will remain.

For example of "Da Ya Wan", we can get 19 Chinese probable characters with pronunciation "Da", 26 Chinese character with pronunciation "Ya", and 29 character with "Wan". In first step, get the occurrence times of all probable associations that pronounced "Da-Ya" (19*26 probable associations), delete the common words, remain the top 5% association by their occurrence times (ex, "打压，打牙，打亚，达雅，达亚，大丫，大压，大鸦，大亚，大呀，塔亚… "). And in second step, get all of the Chinese character sequences of "Da-Ya-Wan" by using the remaining "Da-Ya" associations, get the occurrence times and delete the common words again, remain the best one.

In our experiments, we used the Peoples-Daily corpus of TREC5, because we have not finished our BIG5→GB converter about TREC9 documents when we do this work.

3. Word sense disambiguation.

   We used MI information of two nearby words in queries to do word sense disambiguation. The window is 5 words, and we got the MI information from the segmented Chinese documents supplied by TREC9. The window was 5 words.

   In one of our CLIR run KAIST9xlqm (maximum strategy), we try to select only the Chinese word association that own maximum MI value, if the MI values of given associations are all 0, remain only the first translation in each word. If there are the same MI values between two translation results, remain both of them.

   In KAIST9xlqt run (threshold strategy), we remain all of the Chinese word pairs that own MI values bigger than given threshold 0.

   For example of title CH57, "human right violation" ("human right/f/人权", "violation/n/违犯, 侵犯, 妨碍, 不敬") will be translated as "人权 侵犯" in maximum strategy, and translated to "人权 侵犯, 人权 妨碍" in threshold strategy. And in both strategies, "Chinese press"("Chinese/n/中国 人,中国 话", "press/n/新闻界, 压力 机, 压, 按") will be translated to "中国 人 按" (the correct translation is "中国 新闻界").

4. Double the title field.

   Because the title field includes the most import words or phrases, to improve the IR performance, we double the title field. Our test result shows that this heuristic is quite helpful.

5. Using SMART system to do information retrieving.

3.1.2 Query translation using machine translation system

In our machine translation run KAIST9xlmt, we used two commercial systems by using the combination of the two machine translation results. As the above two runs, we translate the title and description fields of the topics, do segmentation and POS tagging on the translation result, remain only content words (nouns, verbs, adverbs and adjectives), delete some stop words by using our stop word list (this stop word list includes the words that for the description field, like "报告 (report)", "报导 (report)", "文件 (document)"etc.). We can see next section that the result is not so good even after such effect.

3.1.3 Monolingual run

In monolingual run KAIST9xlch, we do nothing except do segmentation to the given Chinese titles and descriptions.

3.1.4 Resources and tools

Resources:

1) Chinese word dictionary with POS information (Yu, 1998), over 50,000 items. Using in Chinese document segmentation, Chinese topic segmentation in two CLIR runs KAIST9xlqm and KAIST9xlqt, and POS Tagging in machine translation run KAIST9xlmt and monolingual run KAIST9xlch.
2) English-Chinese bilingual dictionary with POS information, over 15,000 items. Using in English-Chinese word/phrase translation in two CLIR runs KAIST9xlqm and KAIST9xlqt.
3) Chinese Pinyin-Character table. Using in English-Chinese proper noun translation.
4) Chinese corpus: Peoples-Daily Newspaper that supplied in TREC5 and Chinese documents of TREC9.
5) All of the Chinese resources are in GB code, or converted to GB code from BIG5.

Tools:

1) English Parser of Brill tagger (Brill, 1995): Shareware. Used in our two CLIR run KAIST9xlqm and KAIST9xlqt.
2) SMART Information Retrieval System (Salton, 1983): in all four runs.
3) NJStar Communicator 2.0 (http://www.njstar.com): Shareware, can be download from web. Used as a BIG5→GB code converter.
4) Chinese segmentator and POS Tagger: A part of model of our Chinese-Korean machine translation system (Zhang & Choi, 1999).

3.2 Results

The following table shows the experiment results. We can see the comparison result in the individual queries part, and it based on the average precision over all relevant documents.

Table 3.1 The comparison of the precision.

Total performance				Individual performance				
Run	Avg. Prec.	R-Prec.	Avg. of Median	Best	Above	Median	Below	Worst
KAIST9xlqm	0.2231	0.2145	0.1460	1	10	4	10	0
KAIST9xlqt	0.2107	0.2095	0.1460	1	12	4	7	1
KAIST9xlmt	0.1378	0.1546	0.1460	0	11	1	11	2
KAIST9xlch	0.2233	0.2225	0.2522	4	4	0	16	1

The following is the recall-precision figure on CLIR run KAIST9xlqm (dictionary & corpus based query translation) and KAIST9xlmt (query translation by using machine translation system), we can compare it to the monolingual run KAIST9xlch.

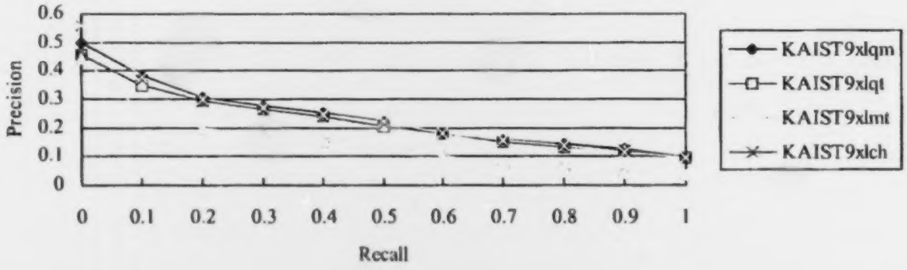


Fig. 3.1 Recall-Precision comparison

In Fig 3.1, we can see that the CLIR run KAIST9xlqm got even high precision than the monolingual run. We think there are several reasons, for example, we only use the noun and noun phrase in our KAIST9xlqm run, but remain all words (even stop words) in monolingual run KAIST9xlch; and we double the title field in our CLIR run KAIST9xlqm while we did not do so in monolingual run, when in the TREC9 topic, the titles reflect the queries very well, and contain only the important words.

The result of the machine translation run KAIST9xlch is not good enough comparing to our exception, especially when it compare to the other CLIR runs. We found that one of the machine translation system generates more noise words and failed to translate almost all of the proper nouns.

Although we pay much attention to the proper noun translation, but the experiment result of the queries that contain proper nouns are under medium yet. We think the reason is that, because the proper nouns not included in our Chinese dictionary, in the Chinese corpus they will be separated to independent characters, and it reflects the information retrieval result directly.

4. Batch Filtering track

We only submitted results for OHSU topics and MSH-SMP topics in batch filtering task. For OHSU topics, we have been exploring a filtering technique which combines query zone (Singhal, 1997), support vector machine (Vapnick, 1995), and Rocchio's algorithm (Rocchio, 1971). For MSH-SMP topics, we use support vector machine simply.

4.1 Experimental Procedure

4.1.1 Profile construction

User profile is created using modified Rocchio's formulation (Rocchio, 1971, Salton, 1990) for each OHSU topic.

$$\vec{P} = \alpha \cdot \vec{Q} + \beta \cdot \frac{1}{R} \sum_{D \in Rel} \vec{D} - \gamma \cdot \frac{1}{N-R} \sum_{D \in Rel} \vec{D} \qquad (4.1)$$

, where $\vec{Q}$ and $\vec{D}$ denote the weighted term vector for query Q and document D, respectively. R = |Rel| is the number of relevant documents, and N is the total number of documents in the collection. The parameters were set to $\alpha=0$, $\beta=1$, and $\gamma=0$.

The weights of terms are calculated by product of term frequency and inverse document frequency.

4.1.2 Filtering based on SVM using Query Zone

We reduced the number of negative training documents for learning of support vector machine using a variation of query zone.

Singhal et al. (Singhal, 1996) have proposed that only a selected set of non-relevant documents that have some relationship to a user's interest should be used in Rocchio's method. They proposed sampling of the non-relevant documents to form a query zone. We selected all documents with similarity to the profile greater than some threshold. If some relevant document does not pass the similarity threshold, it is included in the query zone.

Support vector machines are based on the Structural Risk Minimization principle (Vapnik, 1995) from computational learning theory. The method is defined over a vector space where the problem is to find a decision surface that maximizes the margin between the data points in a training set. We test SVM using the SVM^{light} system (Joachims, 1998) which is an implementation of Vapnik's Support Vector Machine (Vapnik, 1995) for the problem of pattern recognition.

4.1.3 Re-filtering using profile-document similarity

We re-filtered the results from SVM classifier by profile-document similarity. The in-class threshold and out-class threshold are used for re-filtering.

If a profile-document similarity is above in-class threshold, re-filter decide the document to be relevant for user's interest without respect of the result of SVM filter. And, if a profile-document similarity is below out-class threshold, re-filter decide the document to be non-relevant.

4.2 Results

There are three sets of topics: OHSU topics, MSH topics, and MSH-SMP topics. We submitted two runs (KAISTbfo1, KAISTbfo2) for the OHSU topics and one run (KAISTbfms) for MSH-SMP topics.

The TREC-9 filtering track use the OHSUMED collection of documents from MEDLINE. In batch filtering, the 1987 OHSUMED documents are used for building the filtering profiles. The 1988-91 OHSUMED documents form the test set. We didn't use the M. field in the documents for the OHSU topics and MSH-SMP topics.

We tested RBF (radial basis function) models offered by SVM^{light} system. SV learning is based on non-relevant documents from query zone and all relevant documents for each topic. The threshold for query zone was set to 0.1. The number of feature is 31,042. For KAISTbfo1, in-class threshold was set to 0.6 and out-class threshold was set to 0.2. For KAISTbfo2, the thresholds are 0.6 and 0.3. The result of KAISTbfo1 differ little from KAISTbfo2.

Table 4.1 shows the results of OHSU topics and MESH-SAMPLE topics.

Table 4.1 TREC-9 Batch Filtering Results

Topic set Measure	OHSU		MESH-SAMPLE
	KAIST9bfo1	KAIST9bto2	KAIST9bfms
Total retrieved	1615	1437	62483
Relevant retrieved	794	746	35146
Macro average recall	0.227	0.204	0.245
Macro average precision	0.421	0.485	0.543
Mean T9P	0.200	0.194	0.419
Mean utility	12.175	12.714	85.910
Mean T9U	12.175	12.714	86.424
Mean scaled utility	0.061	0.078	0.153
Zero returns	0	2	0

We expected that combined method using QZ, SVM, and Rocchio's algorithm might perform much better than SVM. However, the result of combined method differ little from SVM. A more in-depth analysis is needed to understand these results.

References

Brill, E. (1995) *Transformation-Based error-driven learning and natural language processing: a case study in part of speech tagging*. Computational Linguistics.

Joachims, T. (1998). Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proceedings of the European Conference on Machine Learning (ECML)*.

Miller G.A. & Beckwith R.& Fellbaum C. & Gross D. & Miller K. (1991). Five Papers on WordNet" *International Journal of Lexicography*.

Rocchio, J.J. (1971). Relevance feedback in information retrieval. *In THe SMART Retrieval System - Experiments in Automatic Document Processing*, (pp. 313-323). Prentice Hall, Inc.

Salton, G. & McGill, M.J. (1983*). Introduction to Modern Information Retrieval*, McGraw-Hill, Inc.

Salton, Gerard & Buckley, Chris. (1990). Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4):288-297.

Singhal, Amit & Mitra, & Mandar & Buckley, Chris. 1996. Learning routing queries in a query zone. In *Proceedings of the Twentieth ACM SIGIR Conference on Research and Development in Information Retrieval*, (pp. 21-29).

Vapnick, Vladimir N. (1995). *The Nature of Statistical Learning Theory*, Springer-Verlag, New York.

Yu Shi-Wen, Xue-Feng Zhu, Hui Wang, Yun-Yun Zhang (1998). *The Grammatical Knowledge-base of Contemporary Chinese – A Complete Specification*. The Press of Tsinghua University.

Zhang, Min & Choi, Key-Sun (1999). *Pipelined Multi-Engine Machine Translation : Accomplishment of MATES/CK System*, in Proceedings of the 8th International Conference on Theoretical and Methodological Issues in Machine Translation.

Question Answering
Considering Semantic Categories and Co-occurrence Density

Soo-Min Kim, Dae-Ho Baek, Sang-Beom Kim, Hae-Chang Rim
Dept. of Computer Science and Engineering, Korea University
{smkim,daeho,sbkim,rim}@nlp.korea.ac.kr

Abstract

In this paper, we present a Question Answering system called KUQA (Korea University Question Answering system) developed by using semantic categories and co-occurrence density. Semantic categories are used for computing the semantic similarity between a question and an answer, and co-occurrence density is used for measuring the proximity of the answer to the words of the question. KUQA is developed based on the hypothesis that the words that are semantically similar to the question and locally close to the words appeared in the question are likely to be the answer to the question.

1. Introduction

Question Answering (QA) is defined to find the exact answer to the user's question in a large text collection. In other words, the answer is not the whole document that is relevant to the question, but the parts of the document that can meet the users' need more precisely. On the other hand, current IR systems allow us to locate documents but most of them leave it to the user to extract the information from top ranked documents. Recently, documents have rapidly increased in number, and we need a system that can retrieve *information*, not document. As a result, there has been a growing interest to QA in NLP community.

In this paper, we introduce the KUQA system developed by NLP Lab. in Korea University for the QA track of TREC-9. We try to incorporate NLP techniques with conventional IR techniques. To do this, we utilize WordNet as a kind of linguistic knowledge and a POS tagger for linguistic analyzer.

In the next section, we describe three components of KUQA system. In section 3, we analyze the performance of the system. And finally, we discuss future work in section 4.

2. System Description

Our system consists of three modules: the question analysis module for capturing the meaning of a natural language question, the document retrieval and analysis module for selecting

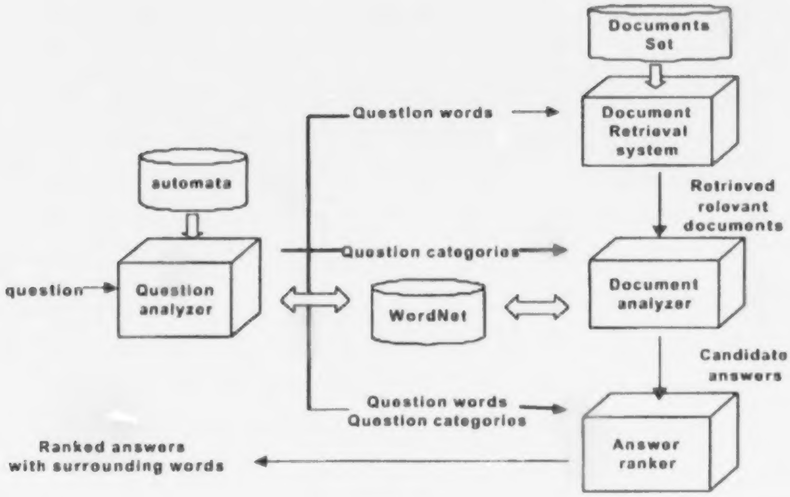

Documents Set
Document Retrieval system
Question words
automata
Retrieved relevant documents
Question categories
question
Question analyzer
WordNet
Document analyzer
Candidate answers
Question words
Question categories
Ranked answers with surrounding words
Answer ranker

Figure 1: Architecture of KUQA system

candidate answers from documents, and the answer extraction module for ranking candidate answers and extracting surrounding words. These modules integrated into KUQA system are represented in Figure1. Each module is described in detail in subsequent sections.

2.1 Question Analysis

The question analyzer reads a question, determines the semantic categories of it by consulting automata and WordNet, and produces a category vector which represents the semantic categories assigned to the question. These classified question categories are used for computing semantic similarity between the question and candidate answers. Also, a list of question words is extracted from the given question, and it is used for retrieving relevant documents and measuring co-occurrence density.

2.1.1. Classifying Question Categories

The question categories indicate the possible type of semantic categories with which the expected answer corresponds. They are decided by different methods according to the types of questions. Questions can be grouped into following three types depending on their interrogatives and sentence structures:

(1) Who, Where, When
(2) How
(3) What, Which, Others

BEST COPY AVAILABLE

question		Question categories
Who		PERSON
Where		COUNTRY CITY CAPITAL PENINSULA ISLAND CONTINENT PROVINCE MOUNTAIN MOUNTAIN_PEAK RIVER OCEAN
When		DAY YEAR TIME_PERIOD TIME_UNIT TIME
How	far , tall	LENGTH LINEAR_UNIT
	long	LENGTH LINEAR_UNIT YEAR TIME TIME_PERIOD TIME_UNIT TIME
	rich	MONETARY_VALUE MONETARY_UNIT ECONOMIC_CONDITION FINANTIAL_LOSS
	much , many	NUMBER
What Which others	Applied to proper automata	The semantic categories of key phrase
	No automata	No category

Table 1: Category assignment table

Table 1 shows the categories assigned to each type of questions. The category of the question with an interrogative *Who, Where, or When* is decided by its meaning of the interrogative. The category of the question with the interrogative *How* is determined by the meaning of an adjective or an adverb followed by the interrogative. For example, *How long* questions may have categories related to time or length and *How many* questions may have categories related to number. However, the categories of *What, Which* and other questions can't be determined just by the meaning of the interrogative or an adjacent adjective or adverb. To analyze these questions, we try to manually construct automata. By using the automata, the system recognizes the key phrase of the question, and then assigns the semantic categories of the question based on the semantic categories of the key phrase. The semantic categories of the key phrase are classified into one of 46 preclassified categories by using WordNet.

Figure 2 shows an example of the process of assigning question categories to the question: *What is the fare cost for the round trip between New York and London on Concord?* In this example, the key phrase "fare cost" is recognized by the automata, and the semantic category of the question is classified into "FINANCIAL LOSS" by using WordNet.

Category vector

Question categories are represented by a *category vector*. The category vector consists of 46 categories manually selected from words in WordNet. If only one category is assigned to a

(Number of categories: 46)

COUNTRY	CITY	PENINSULA	PERSON	LENGTH	......	FINANTIAL_LOSS	MONETARY_UNIT
1	1	1	0	0	......	0	0

Table 2: An example of category vector for the question "*Where is the Orinoco*?"

BEST COPY AVAILABLE

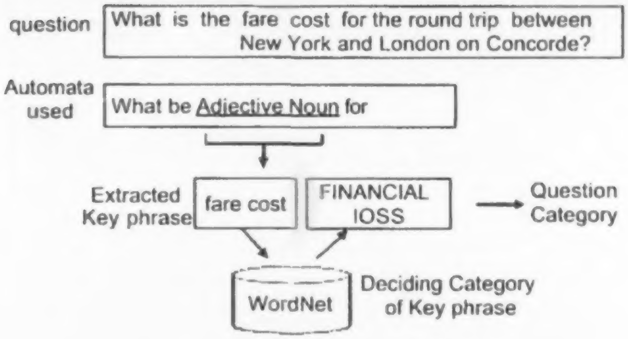

question
What is the fare cost for the round trip between New York and London on Concorde?
Automata used
What be Adjective Noun for
Extracted Key phrase
fare cost
FINANCIAL lOSS
Question Category
WordNet
Deciding Category of Key phrase

Figure2: an example of the process of assigning a question category

question, the value of that category in the category vector is set to 1, and all other categories in the vector are set to 0. If several categories are assigned to a question, then those categories in the vector are set to 1.

Table 2 shows an example of category vector for the question: *Where is the Orinoco?* Since the semantic categories of that question are related to the *location* category, like COUNTRY, CITY, PENINSULA, CONTINENT, and PROVINCE, all the categories related to *location* are set to 1 in the vector.

2.1.2 Extracting Question Words

In the question analysis module, a list of question words is also extracted from the given question, and it is used for retrieving relevant documents and measuring co-occurrence density. A question word is extracted from a question if their part-of-speech is noun, verb, adjective, adverb, or cardinal number, and it is restored to its root form. For example, the list of questions words <fare, cost, round, trip, New, York, London, Concord> is extracted from the question "What is the fare cost for the round trip between New York and London on Concord?"

2.2. Relevant Document Retrieval and Candidate Answer Selection

2.2.1 Retrieving Relevant Documents

The document retrieval module retrieves documents relevant to the question. The document retrieval system is basically implemented based on the OKAPI ranking. However, we assign different weights to the question words according to their part-of-speeches. When the part-of-speech of a question word is proper noun, then the weight of the question word is doubled so that the documents with the same proper noun are highly ranked.

BEST COPY AVAILABLE

2.2.2 Selecting Candidate Answers

We try to select several candidate answers from top 10 ranked documents according to their part-of-speeches.

As shown in table 3, we manually classify semantic categories of a question and their corresponding part-of-speeches. With one or more semantic categories of a question, we can expect the part-of-speeches of candidate answers by using table 3, and then select several candidate answers from top ranked documents based on their part-of-speeches.

In fact, the POS tagger does not tag properly to the word whose part-of-speech is cardinal number. So, we also select the word including a number in its string as candidate answers when the semantic categories of a question expects cardinal number for its candidate answers. For example, the word $600,000 can be a candidate answer when the semantic category of a question is FINANCIAL LOSS, because it contains a number in its string.

COUNTRY, CITY, CAPITAL, PENINSULA, ISLAND, CONTINENT, PROVINCE, MOUNTAIN, MOUNTAIN_PEAK, OCEAN, RIVER, ...	Proper Noun
COMPOUND, MATERIAL, DISEASE, SORT, WORD, BOOK, CINEMA, MOVIE, MUSIC, ...	Proper Noun Noun
NUMBER, LENGTH, LINEAR_UNIT, MAGNITUDE_RELATION, TIME, TIME_PERIOD, YEAR , MONETARY_VALUE, ...	Cardinal Number

Table 3: POS of candidate answers corresponding to semantic categories

2.2.3 Determining Semantic Categories of Candidate answers

We also use WordNet to determine the semantic categories of a candidate answer. Semantic categories of a candidate answer are also represented by a category vector in the same way as the question category vector. The vector consists of 46 categories manually chosen from a pool of words in WordNet.

The system obtains a set of hypernyms and synonyms of a candidate answer by using WordNet. If the set of hypernyms and synonyms of a word contains the words used as categories in the category vector, the system sets the values of those categories in the vector to 1.

Some categories used in the system can be grouped into the classes called *similar category classes*, as shown in table 4. If the category of the candidate answer belongs to one of the class of similar category classes, other categories in the same class are also set to 1 in the vector of that word. For example, the word *cost* has FINANCIAL_LOSS as its hypernym and FINANCIAL_LOSS belongs to one of the similar category classes. Thus, all other categories in the same class: MONETARY_VALUE, MONETARY_UNIT, ECONOMIC_CONDITION are also set to 1.

MOUNTAIN	MOUNTAIN_PEAK
MONETARY_VALUE FINANTIAL_LOSS	MONETARY_UNIT ECONOMIC_CONDITION
TIME TIME_UNIT	TIME_PERIOD
CINEMA	MOVIE
LENGTH	LINEAR_UNIT
WORD	NAME
CAPITAL	CITY

Table 4: Similar category classes

In some cases, two or more words have one category. In the case of words *New York*, although the category of New York is CITY by using WordNet, each word *new* and *York* doesn't belong to the CITY category. To solve this problem, we consider not only the current keyword but also the adjacent words of the current keyword in assigning keyword categories. If there are proper categories for two adjacent words in WordNet, the categories are assigned to the candidate answer.

There are many named entities which are unknown in WordNet. In order to determine the semantic categories of the unknown named entities, we try to use the semantic categories of the adjacent word of the unknown named entity as a clue. As a simple example, consider the phrase: *President Kim said.* The category of the named entity *Kim* can't be determined by using WordNet. But, the category of the preceding word *President* can be determined as PERSON, and the category of unknown named entity *Kim* can be also determined as PERSON.

In the case that several words are connected by hyphens, or a number and a unit together comprise one word, we have to tokenize them as separate words. We divide *92km*, for example, as *92* and *km, and* then determine a category of *92km*. Table 5 shows some examples of words and their corresponding categories.

President Steven	PERSON
New York	CITY
Seoul	CITY
92m	LENGTH, NUMBER
5 may	TIME_PERIOD, NUMBER
$600,000	ECONOMIC_CONDITION FINANCIAL_LOSS MONETARY_VALUE

Table 5: Some examples of words and their categories

2.3 Ranking Candidate Answers

We use three factors to rank candidate answers: average distance weight, co-occurrence ratio, and semantic category similarity. Candidate answers are ranked according to the product of these three factors. By doing that, both semantic category similarity and co-occurrence density are reflected in computing the similarity between a question and an answer.

2.3.1 Average distance weight

By considering the phenomena that an answer to some questions tends to appear in a document locally close to the same words occurred in the question, we use the average distance weight to measure the proximity. Distance weight means the degree of proximity between the keywords of the candidate answer and words in the set of question words. It varies between 0 and 1. Average distance weight is determined by computing the average of distance weight between one candidate word and all question words in the fixed number of words around a candidate answer in a document.

2.3.2 Co-occurrence ratio

Although two candidate answers have the same value of average distance weight, one clustered with many words in the set of question words must have a higher score than the other clustered with just a few words. This is reflected in the following formula of R_i :

$$R_i = \frac{\text{Number of question words appeared in the passage}}{\text{Total Number of question words}}$$

2.3.3 Semantic category similarity

Average distance weight and the co-occurrence ratio are not able to reflect the semantic similarities between a question and a candidate answer. Thus, we define the *category similarity* between a question category vector and a candidate answer category vector. It can have one of three values: high, middle, or low. When two categories are same or similar, the category vector similarity is high. When there is no relevance to each other, the category vector similarity is low.

3. Experimental Results

Our system uses the question set used in TREC8 as a training data. It uses TreeTagger (Helmut Schmid) as a POS tagger and WordNet as a thesaurus. The document retrieval system implemented by using OKAPI algorithm is used for retrieving relevant documents. Our TREC-9 results of 250-byte run are shown in table 6. There are 682 questions in TREC-9 test questions. Unlike the last year, the judgment field can be one of three values: -1 (*Wrong*), 1 (*Correct*), and 2 (*Unsupported*). The Unsupported judgment is given to responses that would have been judged correct but, in the judge's opinion, we could not tell it was a correct answer from the document returned with it.

There are two different evaluations: a strict evaluation which counting only the *Correct* as right and a lenient evaluation that counting both *Correct* and *Unsupported* as right. The first row of the table 6 indicates the result of the strict evaluation and the second row indicates that of the lenient evaluation.

(Total number of test questions: 682)

	Number of answers						MRR	Percentage of correct answers in top 5
	rank1	rank2	rank3	rank4	rank5	Total		
Strict	194	78	35	23	14	344	0.371	50.40%
Lenient	206	74	36	21	16	353	0.386	51.80%

Table6: Result for the 250-byte answer category

4. Discussion and Future Work

In this paper, we introduced our Question-Answering system, named KUQA. With the system, we tried to integrate NLP techniques and IR techniques in a way that makes maximal use of their complimentary ability. KUQA utilizes WordNet as a source of word class information and TreeTagger as a tool for linguistic analysis. Experimental results are encouraging and suggest that NLP techniques are useful for Question-Answering. There is certainly much room for improvement. A problem arises with questions or candidate answers containing words unknown to WordNet. Their semantic categories cannot be classified properly. Another problem arises from limited utilization of NLP techniques. In the future work, we will extend our system to include various NLP techniques including partial parsing, named entity tagging and anaphora resolution.

The LIMSI SDR System for TREC-9

Jean-Luc Gauvain, Lori Lamel, Claude Barras, Gilles Adda, and Yannick de Kercardio

Spoken Language Processing Group (http://www.limsi.fr/tlp)
LIMSI-CNRS, B.P. 133, 91403 Orsay cedex, France
{gauvain,lamel,gadda,barras,kercadio}@limsi.fr

ABSTRACT

In this paper we describe the LIMSI Spoken Document Retrieval system used in the TREC-9 evaluation. This system combines an adapted version of the LIMSI 1999 Hub-4E transcription system for speech recognition with text-based IR methods. Compared with the LIMSI TREC-8 system, this year's system is able to index the audio data without knowledge of the story boundaries using a double windowing approach. The query expansion procedure of the information retrieval component has been revised and makes use of contemporaneous text sources.

Experimental results are reported in terms of mean average precision for both the TREC SDR'99 and SDR'00 queries using the same 557h data set. The mean average precision of this year's system is 0.5250 for SDR'99 and 0.3706 for SDR'00 for the focus unknown story boundary condition with a 20% word error rate.

1. INTRODUCTION

This paper describes the LIMSI broadcast news indexating and retrieval system developed for the TREC-9 Spoken Document Retrieval track. Compared with the LIMSI TREC-8 SDR system, both the speech transcription system and the information retrieval component have been improved. Concerning the speech recognizer, we have both sped up the decoder and slightly reduced the word error rate. The query expansion procedure of the information retrieval component has been revised and the capability to index non-segmented audio streams for the unknown story boundaries condition has been added.

During our developement work we investigated the impact of various system parameters on the IR results including: the transcriber speed, the epoch of the texts used for query expansion, the query expansion term weighting strategy, the query length, and the use of non-lexical information.

Most of the reported results here were obtained using the TREC-8 SDR'99 conditions, i.e. the TREC-8 data collection consisting of 557 hours of broadcast news from the period of February through June 1998. This data includes 21750 stories and has an associated set of 50 queries.

The remainder of this paper is as follows: In the next three sections we provide an overview of the broadcast news indexation and information retrieval components, followed by an investigation of the impact of decoding speed and the consequence of the word error rate on the information retrieval process. The subsequent two sections address query expansion and the use of non-lexical information. We then describe how we addressed the unknown story boundary condition and the terse query condition in this year's evaluation. Comparative results are given on the development queries from SDR'99 and this year's query set, and some conclusions are made.

2. TRANSCRIPTION SYSTEM OVERVIEW

The LIMSI broadcast news transcription system [5] consists of an audio partitioner [10] and a speech recognizer [11, 12]. The goal of audio partitioning is to divide the acoustic signal into homogeneous segments, labeling and structuring the acoustic content of the data. Partitioning consists of identifying and removing non-speech segments, and then clustering the speech segments and assigning bandwidth and gender labels to each segment. The result of the partitioning process is a set of speech segments with cluster, gender and telephone/wideband labels, which can be used to generate metadata annotations. The partitioning approach used in the LIMSI BN transcription system relies on an audio stream mixture model [10]. Each component audio source, representing a speaker in a particular background and channel condition, is modeled by a GMM. The segment boundaries and labels are jointly identified by an iterative maximum likelihood segmentation/clustering procedure using GMMs and agglomerative clustering.

For each speech segment, the word recognizer determines the sequence of words in the segment, associating start and end times and an optional confidence measure with each word. The speaker-independent large vocabulary, continuous speech recognizer makes use of n-gram statistics for language modeling and of continuous density HMMs with Gaussian mixtures for acoustic modeling. Word recognition is usually performed in three steps: 1) initial hypothesis generation, 2) word graph generation, 3) final hypothesis generation. The hypotheses are used in cluster-based acoustic model adaptation using the MLLR technique [16] prior to word graph generation, and all subsequent decoding passes. The final hypothesis is generated using a 4-gram language

BEST COPY AVAILABLE

model.

For all the experimental results given in this paper, the following training conditions were used. The acoustic models were trained on about 150 hours of American English broadcast news data. The phone models are position-dependent triphones, with about 11500 tied-states for the largest model set. The state-tying is obtained via a divisive, decision tree based clustering algorithm. Wideband and telephone band sets of gender-dependent acoustic models were built using MAP adaptation of SI seed models. Fixed language models were obtained by interpolation of n-gram backoff language models trained on 3 different data sets: 203 M words of BN transcripts; 343 M words of NAB newspaper texts and AP Wordstream texts; 1.6 M words corresponding to the transcriptions of the acoustic training data. The interpolation coefficients of these LMs were chosen so as to minimize the perplexity on the Hub4 Nov98 evaluation data. The 4-gram LM contains 7M bigrams, 14M trigrams and 11M fourgrams.

The recognition word list contains 65122 words. The word pronunciations are based on a 48 phone set (3 of them are used for silence, filler words, and breath noises). A pronunciation graph is associated with each word so as to allow for alternate pronunciations, including optional phones. Frequent inflected forms have been verified to provide more systematic pronunciations. As done in the past, compound words for about 300 frequent word sequences subject to reduced pronunciations were included in the lexicon as well as the representation of the most frequent acronyms as words.

3. INFORMATION RETRIEVAL

The automatically generated partition and word transcription can be used for indexation and information retrieval purposes. Techniques commonly applied to automatic text indexation were applied to the automatic transcriptions of the broadcast news radio and TV documents. These classical techniques are based on document term frequencies, where the terms are obtained after standard text processing, such as text normalization, tokenization, stopping, stemming and named-entity identification.

In order to be able to apply the same IR system to different text data types (automatic transcriptions, closed captions, additional texts from newspapers or newswires), all of the documents are preprocessed in a homogeneous manner. This preprocessing, or tokenization, is the same as the text source preparation for training the speech recognizer language models [7], and attempts to transform the texts to be closer to the observed American speaking style. The basic operations include translating numbers and sums into words, removing all the punctuation symbols, removing case distinctions and detecting acronyms and spelled names. However removing all punctuations implies that certain hyphenated words such as *anti-communist*, *non-profit* are rewritten as *anti communist* and *non profit*. While this offers advantages for speech recognition, it can lead to IR errors. To avoid IR problems due to this type of transformation, the output of the tokenizer (and recognizer) is checked for common prefixes, in order to rewrite a sequence of words such as *anti communist* as a single word. The prefixes that are handled include *anti*, *co*, *bi*, *counter*. A rewrite lexicon containing compound words formed with these prefixes and a limited number of named entities (such as *Los-Angeles*) is used to transform the texts. Similarly all numbers less than one hundred are treated as a single entity (such as *twenty-seven*).

In order to reduce the number of lexical items for a given word sense, each word is translated into its stem (as defined in [2, 21]) or, more generally, into a form that is chosen as being representative of its semantic family. The stemming lexicon (derived from the UMass 'porterized' lexicon) [2] contains about 32000 entries and was constructed using Porter's algorithm on the most frequent words in the collection, and then manually corrected.

Two approaches for IR were explored for SDR'99 and this year, the first based on the popular TF*IDF weigthing scheme and the second using a Markovian term weighting [14, 17, 19].

For the TF*IDF approach, the score of document d for a query is given by the Okapi-BM25 formula[22, 23]. It is the sum over all the terms t in the query of:

$$cw_{t,d} = \frac{(K+1) * tf_{t,d}}{K * (1 - b + b * L_d) + tf_{t,d}} * \log \frac{N}{N_t} * qtf_t$$

where $tf_{t,d}$ is the number of occurrences of term t in document d (i.e. term frequency in document), N_t is the number of documents containing term t at least once, N is the total number of documents in the collection, L_d is the length of document d divided by the average length of the documents in the collection, and qtf_t the number of occurrences of term t in the query.

For the second approach the score of a story is obtained by summing the query term weights $mw_{t,d}$ which are the unigram log probabilities of the terms given the story model once interpolated with a general English model:

$$mw_{t,d} = qtf_t * \log(\alpha \Pr(t|d) + (1 - \alpha) \Pr(t)).$$

The text of the query may or may not include the index terms associated with relevant documents. One way to cope with this problem is to use query expansion based on terms present in retrieved documents on the same (Blind Relevance Feedback, BRF) or other (Parallel Blind Relevance Feedback, PBRF) data collections [24]. For SDR'99 we combined the two approches in our system. For PBRF we used 6 months of commercially available broadcast news transcripts from the period jun-dec 1997 [1]. This corpus contains 50000 stories and 49.5 M words. For a given query, the terms found in the top B documents from the baseline search are ranked by their *offer weight* [23], and the top T terms are added to the query. Since only the T terms with best offer weights are kept, the terms are filtered using a stop list of 144 common words, in order to increase the likelihood that the resulting terms are relevant.

Table 1 gives the results for both *cw* and *mw* term weightings for the SDR'99 data set. Four experimental configurations are reported: baseline search (*base*), query expansion using BRF (*brf*), query expansion with parallel BRF (*pbrf*) and query expansion using both BRF and PBRF (*brf+pbrf*). For BRF and PBRF, the terms are added to the query with a weight of 1. For BRF+PBRF, the terms from each source are added with a weight of 0.5. The results clearly demonstrate the interest of using both BRF and PBRF expansion techniques, as consistent improvements are obtained over the baseline system for the two conditions (R1 and S1). BRF is found to be more effective for both the S1 condition (the recognizer transcripts) and the R1 condition (the manual transcripts).

data	*meth.*	*base*	*brf*	*pbrf*	*brf+pbrf*
R1K	*tf*idf*	0.4711	0.5318	0.5147	0.5487
	unigram	0.4691	0.5354	0.5098	0.5430
S1K	*tf*idf*	0.4327	0.5239	0.4919	0.5350
	unigram	0.4412	0.5302	0.4943	0.5398

Table 1: Comparison of IR results on the SDR'99 data set using both Okapi and Markovian term weightings (b=0.86, K=1.1, B=15, T=10, α=0.5). R1: reference transcript. S1: automatic speech transcription. K: known story boundary condition.

The two IR approaches are seen to yield comparable results [13]. Only small differences in information retrieval performance as given by the mean average precision were observed for automatic and manual transcriptions when the story boundaries are known.

4. DECODING SPEED

Processing time is an important factor in making a speech transcription system viable for automatic indexation of radio and television broadcasts. When only concerned by the word error rate, it is common to design systems that run in 100 times real-time or more. Although it is usually assumed that processing time is not a major issue since computer power has been increasing continuously, it is also known that the amount of data appearing on information channels is increasing very rapidly. Therefore processing time is an important factor in making a speech transcription system viable for audio data mining and other related applications. Constraints on the computational resources led us to reconsider some of the system design issues, particularly those concerning the acoustic models and the decoding strategy. We investigated the design of a system which performs well with computational resources in the range 1 to 10xRT on commonly available platforms. A new decoder was implemented with which broadcast data can be transcribed in few times real-time with only a slight increase in word error rate when compared to our best system.

A 4-gram single pass dynamic network decoder has been developed. It is a time-synchronous Viterbi decoder with dynamic expansion of LM state conditioned lexical trees [3, 18, 20] with acoustic and language model lookaheads. The decoder can handle position-dependent, cross-word triphones and lexicons with contextual pronunciations. It makes use of various pruning techniques to reduce the search space and computation time, including three HMM-state pruning beams and fast Gaussian likelihood computations. It can also generate word graphs and rescore them with different acoustic and language models. Faster than real-time decoding can be obtained using this decoder with a word error under 30%, running in less than 100 Mb of memory on widely available platforms such Pentium III or Alpha machines.

The decoder by itself does not solve by itself the problem of reducing the recognition time as proper models have to be used in order to optimize the recognizer accuracy at a given decoding speed. In general, better models have more parameters, and therefore require more computation. However, since the models are more accurate, it is often possible to use a tighter pruning level (thus reducing the computational load) without any loss in accuracy. Thus, limitations on the available computational resources affect the design of the acoustic and language models. For each operating point, the right balance between model complexity and pruning level must be found.

In order to assess the effect of the recognition time on the information retrieval results we transcribed the 557 hours of broadcast news data (the TREC SDR'99 data set – epoch Feb98 to Jun98) using two decoder configurations: a single pass 1.4xRT system and a three pass 10xRT system. The SDR'99 test data consists of 21750 stories and an associated set of 50 queries with on average 14 words. Although story boundaries are available, this information is not used by the speech recognizer. The information retrieval results are given in term of mean average precison (MAP), as is done for the TREC benchmarks. Word error rates are measured on a 10h test subset [6]. For comparison, results are also given for manually produced closed captions. In order for the same IR system to be applied to different text data types (automatic transcriptions, closed captions, additional texts from newspapers or newswires), all of the documents are preprocessed in a homogeneous manner. This preprocessing, or tokenization, is the same as the text source preparation for training the speech recognizer language models.

Table 2 gives the word error rates and IR results for the three sets of transcriptions with and without query expansion. Query expansion uses blind relevance feedback (BRF)

Transcriptions	*Werr*	*Base*	*BRF*
Closed-captions	-	0.4691	0.5430
10xRT	20.5%	0.4528	0.5385
1.4xRT	32.6%	0.4090	0.4938

Table 2: Impact of the word error rate on the mean average precision using the SDR'99 conditions using a 1-gram document model.

pbrf'99	*brf+pbrf'99*	*pbrf'00*
0.5017	0.5397	0.5956

Table 3: Comparison of query expansion schemes on the SDR'99 data with known story boundaries.

on both the audio document collection and some commercially available broadcast news transcripts predating the audio corpus (Jun-Dec 1997 vs Feb-Jun 1998). With query expansion comparable IR results are obtained using the closed captions and the 10xRT transcriptions, and a small degradation (4% absolute) is observed using the 1.4xRT transcriptions.

5. QUERY EXPANSION

In our SDR'99 system query expansion was done by adding terms present in retrieved documents on the same data collection and in an independent set of texts. For PBRF we made use of 6 months of commercially available broadcast news transcripts for covering the period of June through December 1997 [1] (50000 stories and 49.5 M words). However, the SDR'00 specifications (as well as the SDR'99 specifications) allow us to use texts (except for BN transcripts) covering exactly the same epoch of the audio data. Therefore this year we implemeted PBRF using 3 sources of contemporary newspaper data: the New York Times, the Los Angeles Times and the Washington Post. The parallel corpus conatined a total of 42 M words and 78 K documents between Jan98 and Jun98. Experiments with these texts on the SDR'99 show that PBRF using contemporary texts offers a significant performance gain compared with a PBRF using texts predating the audio data. In fact we found that we no longer needed to combine both BRF and PBRF, since PRBF with the new texts gave comparable benefits.

This year we also changed the term weighting used with query expansion, using a weight proportional to the *offer weight* as defined in [23, 15]. This approach allowed us to significantly increase the number of expansion terms, going from 10 terms with the previous approach to 25 terms with the term weighting. The sum of the weights for the expansion terms is set to the number of added terms, i.e., 25. Table 3 shows the combined improvment obtained with the new query expansion scheme on the SDR'99 data. These results were obtained using the Okapi term weighting with a parameter setting (b=0.7, K=1.2) and a slighlty different stemmer from that used for the results reporter earlier in this paper.

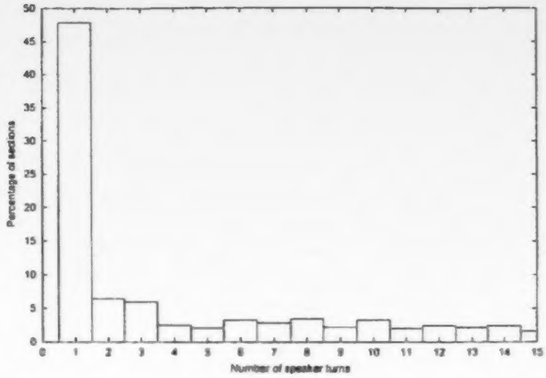


Figure 1: Histogram of the number of speaker turns per section in the 1997 Hub-4 data set.

6. NON-LEXICAL INFORMATION

The broadcast news transcription system also provides non-lexical information along with the word transcription. This information is available in the partition of the audio track, which identifies speaker turns. We investigated the use of automatically detected speaker changes for locating document boundaries. Statistics were made on the 1997 English Hub-4 training data set, which consists of about 100 hours of radio and television broadcast news with manual transcription and speaker identification. On this set, 2096 sections were manually marked as report sections and used as documents for the SDR'98 evaluation. Among them, 817 sections (about 40%) start without a manually annotated speaker change. This means that using only speaker change information for detecting document boundaries would result in 40% missed boundaries. This figure would likely increase with the use of automatically detected speaker changes. At the same time, 11,160 of the total of 12,439 speaker turns occur in the middle of a document, which gives almost a 90% false alarm rate. A more detailed analysis shows that about 50% of the sections involve a single speaker, but that the distribution of the number of speaker turns per section falls off very gradually from 2 to 20 speakers (cf. Figure 1). False alarms are not as harmful as missed detections, since it is possible to merge adjacent turns into a single document in subsequent processing. However these results show clearly that even perfect speaker turn boundaries cannot be used as the primary cue for locating document boundaries. They can be used to refine the placement of a document boundary located near a speaker change.

Besides speaker turns, changes in the background acoustic conditions can be detected by the audio partitioner and can be considered as indicators of story boundaries. We did not investigate this because the background conditions were not manually marked in the 1997 English Hub-4 corpus.

We investigated using simple statistics on the durations of

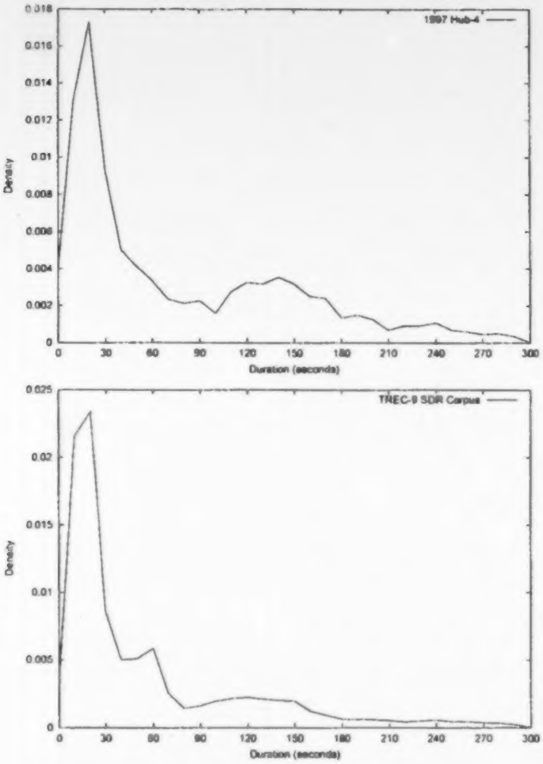


Figure 2: Distribution of document durations in the Hub4'97 and SDR'00 data sets.

the documents in the SDR'98 data set. A histogram of the 2096 sections is shown in Figure 2. One third of the sections are shorter than 30 seconds. The histogram has a sharp peak around 20 seconds, and a smaller, flat peak around 2 minutes, resulting in a bimodal distribution of document length. Very short documents are typical of headlines which are uttered by single speaker, whereas longer documents are more likely to contain data from multiple talkers. This distribution led us to consider using a multi-scale segmentation of the audio stream into documents. Similar statistics were measured on the SDR'99 data using the known document boundaries. The distribution, shown in lower part of Figure 2 is quite similar to that of the SDR'98 data, with an additional, small peak at 60 seconds.

7. UNKNOWN STORY BOUNDARY CONDITION

As proposed in [9], we first segmented the audio stream into overlapping documents of a fixed duration. As a result of optimization using the TREC-8 SDR queries, we chose a 30 second window duration with a 15 second overlap. Since there are many stories significantly shorter than 30s in broadcast shows (see Figure 2) we conjunctured that it may be of interest to use a double windowing system in order to better target short stories. The window size of the smaller window was selected to be 10 seconds. So for each query, we independently retrieved two sets of 2700 documents, one set for each window size. Then for each document set, document recombination is done by merging overlapping documents until no further merges are possible. The score of a combined document is set to maximum score of any one of the components. For each document derived from the 30s windows, we produce a time stamp located at the center point of the document. However, if any smaller documents are embedded in this document, we take the center of the best scoring document. This way we try to take avantage of both window sizes. The MAP using a single 30s window and the double windosing strategy are shown in Table 4.

Mode	*30s*	*30s + 10s*
baseline	0.3673	0.3791
PBRF	0.5001	0.5260

Table 4: Unkown story boundary condition development results on SDR'99 data.

8. TERSE QUERIES

A new component of this year's evaluation was the use of terse queries for indexation. Since terse forms of the 1999 queries were not available, we generated a set for use in system development. These were generated based on the instructions given to the assessors that developed the SDR'00 short and terse queries.[1] Different group members used these general instructions to independently generate terse versions of the SDR'99 queries. These were then compiled and a single form was selected. The resulting SDR'99 terse queries contain on average 3.3 words per query to be compared to 13.7 words for the regular "short" queries.

We carried out retrieval experiments with these terse queries using the system parameter values tuned for the short queries. The retrieval results are given on Table 5 for both the known and unknown story boundary conditions on the SDR'99 data. We can see that there is only about a 1% absolute reduction of the mean average precision when the short queries are replaced by the terse queries. Given this small degradation we did not try to modify our system to better optimize performance on the terse queries.

9. RESULTS

Retrieval results for the SDR'00 evaluation system are given in Tables 6 and 7 for both SDR'99 and SDR'00 queries. It is clear from these results that the system behavoir is quite different on the two query sets. First the SDR'00

---

[1] Although no specific written guidelines were available, John Garofolo kindly described the instructions given to the assessors.

Mode	*short queries*	terse queries
R1K	0.5975	0.5852
S1K	0.5956	0.5795
S1U	0.5260	0.5147

Table 5: Retrieval results with short and terse queries on the SDR'99 data. R1: reference transcript. S1: automatic speech transcription. K: known story boundary condition. U: unknown story boundary condition.

queries appear to be significantly more difficult, with a 25% relative reduction in the mean average precision compared to the SDR'99 queries. Second, we get significantly better results with the terse queries than with the short queries, while we observed a slight loss on our SDR'99 terse queries. The average length of the SDR'00 terse queries (3.0) is not significantly different from the average length of our SDR'00 terse queries (3.3), but there is a substantial difference in the number of new words compared to the short queries. The SDR'00 terse queries introduce 54 new words with 85 words in common the the SDR'00 short queries, whereas we had only 17 new words in our SDR'99 terse queries with 181 words in common. These numbers show that our SDR'99 terse queries were essentially shorter versions of the corresponding short query, whereas the SDR'00 terse queries appear to be a reformulation of the SDR'00 short queries.

Mode	*Queries'99*		*Queries'00*	
	short	*terse*	*short*	*terse*
R1K	0.5975	0.5852	0.4636	0.5132
S1K	0.5956	0.5795	0.4327	0.4812

Table 6: Retrieval results on SDR'99 and SDR'00 data with known story boundaries. R1: reference transcript. S1: automatic speech transcription. K: known story boundary condition.

Mode	*Queries'99*		*Queries'00*	
	short	*terse*	*short*	*terse*
R1U	0.5233	-	0.4027	0.4283
B1U	0.5034	-	0.3712	0.3922
S1U	0.5260	0.5147	0.3706	0.3982

Table 7: Retrieval results on SDR'99 and SDR'00 data with unknown story boundaries. R1: reference transcript. B1: baseline automatic speech transcription. S1: automatic speech transcription. U: unknown story boundary condition.

10. CONCLUSION

In this paper we have described the LIMSI TREC-9 spoken document retrieval system. This system is based on the 1999 LIMSI system, with a few substantial modifications. First, the decoder of the speech recognizer has been replaced by a new, faster decoder able to transcribes broadcast data in several (6 to 10) times real-time with only a slight increase in word error rate when compared to our best system and with a word error of about 30% for essentially real-time decoding. Second, the query expansion procedure of the information retrieval component has been revised and makes use of contemporaneous text sources. Thirdly, a double windowing approach has been developed to localize stories for the unknown boundary condition.

The experimental results show that only a moderate IR performance degradation is obtained in spoken document retrieval with a close to real-time system, and that generally speaking, the transcription quality of our system is not a limiting factor given todays IR techniques.

ACKNOWLEDGEMENTS

This work has been partially financed by the European Commission under the IST-1999-10354 ALERT project and the French Ministry of Defense. We also thank Patrick Paroubek for providing the terse versions of the SDR'99 query set used for system development.

REFERENCES

[1] http://www.thomson.com/psmedia/bnews.html

[2] ftp://ciir-ftp.cs.umass.edu/pub/stemming/

[3] X. Aubert, "One Pass Cross Word Decoding for Large Vocabularies Based on a Lexical Tree Search Organization," *Proc. ESCA Eurospeech'99*, **4**, pp. 1559-1562, Budapest, Hungary, September 1999.

[4] J. Davenport, L. Nguyen, S. Matsoukas, R. Schwartz, J. Makhoul, "The 1998 BBN Byblos 10x Real Time System," *Proc. DARPA Broadcast News Workshop*, Feb.-Mar. 1999.

[5] J.L. Gauvain, L. Lamel, and G. Adda, "Transcribing broadcast news for audio and video indexing," Communications of the ACM, 43(2), February 2000.

[6] J.S. Garofolo et al., "1999 Trec-8 Spoken Document Retrieval Track Overview and Results," *Proc. 8th Text Retrieval Conference TREC-8*, Gaithersburg, MD, November 1999.

[7] J.L. Gauvain, L. Lamel, M. Adda-Decker, "The LIMSI Nov93 WSJ System," *Proc. ARPA Spoken Language Technology Workshop*, March, 1994.

[8] J.L. Gauvain, Y. de Kercadio, L.F. Lamel, G. Adda "The LIMSI SDR system for TREC-8," *Proc. of the 8th Text Retrieval Conference TREC-8*, pp. 405-412, Gaithersburg, MD, November 1999.

[9] D. Abberley, S. Renals, Dan Ellis and T. Robinson, "The THISL SDR System at TREC-8", *Proc. of the 8th Text Retrieval Conference TREC-8*, Gaithersburg, MD, November 1999.

[10] J.L. Gauvain, L. Lamel, G. Adda, "Partitioning and Transcription of Broadcast News Data," *ICSLP'98*, **5**, pp. 1335-1338, December 1998.

[11] J.L. Gauvain, L. Lamel, G. Adda and M. Jardino, "The LIMSI 1998 Hub-4E Transcription System", *Proc. DARPA Broadcast News Workshop*, pp. 99-104, Herndon, VA, February 1999.

[12] J.L. Gauvain, L. Lamel, G. Adda, "Recent Advances in Transcribing Television and Radio Broadcasts," *Proc. Eurospeech'99*, **2**, pp. 655-658, Budapest, Hungary, September 1999.

[13] J.L. Gauvain, L. Lamel, Y. de Kercadio, and G. Adda, "Transcription and Indexation of Broadcast Data," *Proc. IEEE ICASSP'00*, **III**, pp. 1663-1666, Istanbul, Turkey, June 2000.

[14] D. Hiemstra, K. Wessel, "Twenty-One at TREC-7: Ad-hoc and Cross-language track," *Proc. of the 8th Text Retrieval Conference TREC-7*, Gaithersburg, MD, 1998.

[15] S.E. Johnson, P. Jourlin, K. Spärck Jones, P.C. Woodland, "Spoken Document Retrieval for TREC-8 at Cambridge University", *Proc. of the 8th Text Retrieval Conference TREC-8*, Gaithersburg, MD, November 1999.

[16] C.J. Leggetter, P.C. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models," *Computer Speech & Language*, **9**(2), pp. 171-185, 1995.

[17] D. Miller, T. Leek, R. Schwartz, "Using Hidden Markov Models for Information Retrieval", *Proc. of the 8th Text Retrieval Conference TREC-7*, Gaithersburg, MD, 1998.

[18] H. Ney, R. Haeb-Umbach, B.H. Tran and M. Oerder, "Improvements in Beam Search for 10000-Word Continuous Speech Recognition," *Proc. IEEE ICASSP-92*, **I**, pp. 9-12, San Francisco, CA, March 1992.

[19] K. Ng, "A Maximum Likelihood Ratio Information Retrieval Model," *Proc. of the 8th Text Retrieval Conference TREC-8*, pp. 413-435, Gaithersburg, MD, November 1999.

[20] J.J. Odell, V. Valtchev, P.C. Woodland and S.J. Young, "A One Pass Decoder Design for Large Vocabulary Recognition," *Proc. ARPA Human Language Technology Workshop*, pp. 405-410, Princeton, NJ, March 1994.

[21] M. F. Porter, "An algorithm for suffix stripping", *Program*, **14**, pp. 130–137, 1980.

[22] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, "Okapi at TREC-3", *NIST Special Publication 500-226: Overview of the Third Text REtrieval Conference (TREC-3)*, November 1994.

[23] K. Spärk Jones, S. Walker, S. E. Robertson, "A probabilistic model of information retrieval: development and status", *a Technical Report of the Computer Laboratory, University of Cambridge, U.K.*, 1998.

[24] S. Walker, R. de Vere, "Improving subject retrieval in online catalogues: 2. Relevance feedback and query expansion", *British Library Research Paper 72*, British Library, London, U.K., 1990.

[25] P.C. Woodland, J.J. Odell, T. Hain, G.L. Moore, T.R. Niesler, A. Tuerk, E.W.D. Whittaker, "Improvements in Accuracy and Speed in the HTK Broadcast News Transcription System," *Eurospeech'99*, pp. 1043-1046, September 1999.

The TREC-9 Filtering Track Final Report

Stephen Robertson
Microsoft Research
Cambridge, UK
ser@microsoft.com

David A. Hull
WhizzBang Labs
Pittsburgh PA, USA
dhull@whizbang.com

Abstract

The TREC-9 filtering track measures the ability of systems to build persistent user profiles which successfully separate relevant and non-relevant documents. It consists of three major subtasks: adaptive filtering, batch filtering, and routing. In adaptive filtering, the system begins with only a topic statement and a small number of positive examples, and must learn a better profile from on-line feedback. Batch filtering and routing are more traditional machine learning tasks where the system begins with a large sample of evaluated training documents. This report describes the track, presents some evaluation results, and provides a general commentary on lessons learned from this year's track.

1 Introduction

A text filtering system sifts through a stream of incoming information to find documents relevant to a set of user needs represented by profiles. Unlike the traditional search query, user profiles are persistent, and tend to reflect a long term information need. With user feedback, the system can learn a better profile, and improve its performance over time. The TREC filtering track tries to simulate on-line time-critical text filtering applications, where the value of a document decays rapidly with time. This means that potentially relevant documents must be presented immediately to the user. There is no time to accumulate and rank a set of documents. Evaluation is based only on the quality of the retrieved set.

Filtering differs from search in that documents arrive sequentially over time. The TREC filtering track consists of three subtasks: adaptive filtering, batch filtering, and routing. In adaptive filtering, the system starts with only a user profile and (in TREC-9) a very small number, two or four, of positive examples (relevant documents). It must begin filtering documents without any other prior information. Each retrieved document is immediately judged for relevance, and this information can be used by the system to adaptively update the filtering profile. In batch filtering and routing, the system starts with a large set of evaluated training documents which can be used to help construct the search profile. For batch filtering, the system must decide to accept or reject each document, while routing systems can return a ranked list of documents. The core tasks have remained the same in TREC-7 through TREC-9.

Traditional adhoc retrieval and routing simulate a non-interactive process where users look at documents once at the end of system processing. This allows for ranking or clustering of the retrieved set. The filtering model is based on the assumption that users examine documents periodically over time. The actual frequency of user interaction is unknown and task-dependent. Rather than create a complex simulation which includes partial batching and ranking of the document set, we make the simplifying assumption that users want to be notified about interesting documents as soon as they arrive. Therefore, a decision must be made about each document without reference to future documents, and the retrieved set is ordered by time, not estimated likelihood of relevance. The history and development of the TREC Filtering Track can be traced by reading the yearly final reports:

- TREC–8 http://trec.nist.gov/pubs/trec8/t8_proceedings.html (#3 - 2 files) [5]
- TREC–7 http://trec.nist.gov/pubs/trec7/t7_proceedings.html (#3 - 2 files) [4]
- TREC–6 http://trec.nist.gov/pubs/trec6/t6_proceedings.html (#4 and #5) [3]
- TREC–5 http://trec.nist.gov/pubs/trec5/t5_proceedings.html (#5) [7]
- TREC–4 http://trec.nist.gov/pubs/trec4/t4_proceedings.html (#11) [6]

Information on the participating groups and their filtering systems can be found in the individual site reports, also available from the TREC web site.

2 TREC–9 Task Description

The basic filtering tasks in TREC–9 have not changed from TREC–8, nor even from TREC–7, with two exceptions. These are: (a) the provision of a few positive training examples for the adaptive filtering task; and (b) the introduction of a new evaluation measure. The corpus and topics are also somewhat different from those used previously. In this section, we review the corpus, the three sub-tasks, the submission requirements, and the evaluation measures. For more background and motivation, please consult the TREC–7 track report [4].

2.1 Data

The TREC–9 filtering experiments went outside the usual TREC collections and used a slightly modified version of the OHSUMED test collection compiled by, and available from, Bill Hersh [1]. This consists of Medline documents from the years 1987–1991 and a set of requests (topics) and relevance judgements. The modified dataset used for filtering may be described as follows.

The entire collection contains about 350,000 documents. Actually these are bibliographic records containing the usual fields including abstract, although only about two thirds of the records contain abstracts. They also have a field containing MeSH headings, that is human-assigned index terms. These are assumed to arrive in identifier order, at a rate of approximately 6000 documents per month. The 1987 data (equivalent to about 9 months' worth) was extracted from the dataset to provide training material, as discussed below; the test set is therefore the 1988–91 data.

Sixty-three of the original OHSUMED topics were selected for filtering (they were selected to have a minimum of 2 definitely relevant documents in the training set)[1]. These 63 topics form the OHSU set. In addition, the MeSH headings were treated as if they were topics: the text of the topic was taken from the scope notes available for MeSH headings, and assignments of headings to documents were regarded as relevance judgements. Again they were selected, to have a minimum of 4 relevant documents in the training set and to have at least one in the final year; also very rare and very frequent headings were excluded.[2] The remaining 4903 MeSH headings formed the MSH topic set. Finally, because of the size of this topic set which made it difficult to process in its entirety, a random sample of 500 of these was made, to form the MSH-SMP set.

---

[1] Relevance judgements for OHSUMED topics were made on a 3-point scale, not relevant, possibly relevant and definitely relevant. The training documents for adaptive filtering were definitely relevant. Systems were free to make use of the graded relevance judgements in any way they saw fit, but the final evaluation was based on treating both possibly relevant and definitely relevant as relevant.

2.2 Tasks

The adaptive filtering task is designed to model the text filtering process from the moment of profile construction. In TREC–9, in contrast to previous TRECs, we model the situation where the user arrives with a small number of known positive examples (relevant documents). Subsequently, once a document is retrieved, the relevance assessment (when one exists) is immediately made available to the system. Unfortunately, it is not feasible in practice to have interactive human assessment by NIST. Instead, assessment is simulated by releasing the pre-existing relevance judgement for that document. Judgements for unretrieved documents are never revealed to the system. Once the system makes a decision about whether or not to retrieve a document, that decision is final. No back-tracking or temporary caching of documents is allowed. While not always realistic, this condition reduces the complexity of the task and makes it easier to compare performance between different systems.

Systems are allowed to use the whole of the training set of 1987 documents to generate collection frequency statistics (such as IDF) or auxiliary data structures (such as automatically-generated thesauri). Resources outside the OHSUMED collection could also be used, as could the OHSUMED topics and MeSH headings excluded from the filtering task, with all relevance judgements on the 1987 documents, for system training. As documents were processed, the text could be used to update term frequency statistics and auxiliary document structures even if the document was not matched to any profile. Groups had the option to treat unevaluated documents as not relevant.

In batch filtering, all 1987 documents and all relevance judgements on that set of documents are available in advance as a training set. The 1988-91 documents form the test set. As in adaptive filtering, systems may use the relevance judgement from any retrieved document to update the filtering profile (if these are used it becomes batch-adaptive filtering). For routing, the training data is the same as for batch filtering; systems return a ranked list of the top 1000 retrieved documents from the 1988-91 set. Batch filtering and routing are included to open participation to as many different groups as possible.

2.3 Evaluation and optimisation

For the TREC experiments, filtering systems are expected to make a binary decision to accept or reject a document for each profile. Therefore, the retrieved set consists of an unranked list of documents. This fact has implications for evaluation, in that it demands a measure of effectiveness which can be applied to such an unranked set. Many of the standard measures used in the evaluation of ranked retrieval (such as average precision) are not applicable. Furthermore, the choice of primary measure of performance will impact the systems in a way that does not happen in ranked retrieval. While good ranking algorithms seem to be relatively independent of the evaluation measure used, good classification algorithms need to relate very strongly to the measure it is desired to optimise.

Two measures were used in TREC–9 for this purpose (as alternative sub-tasks). One was essentially the linear utility measure used in previous TRECs, and described below. The other was new for TREC–9, and is described as a precision-oriented measure.

---

[2] The reason for excluding those MeSH headings not represented in the final year was to avoid headings which had been dropped out of MeSH (which undergoes continual modification) during the period.

Precision-oriented measure

The idea of this measure is to set a target number of documents to be retrieved over the period of the simulation; the target was set at 50 documents for each topic (the same for all topics). This situation might be said to correspond roughly with cases where the user indicates what sort of volume of material they expect / are prepared for / are able to deal with / would like to see. Clearly a fixed target is a simplification of such cases (each of which is a little different from the others), but may be seen as an acceptable simplification for experimental purposes.

The measure is essentially precision, but with a penalty for not reaching the target:

```
          Number of relevant retrieved documents
T9P = ------------------------------------------
          Max(Target, Number of retrieved documents)

Target = 50 documents
```

This may be regarded as something akin to a "precision at [target] documents" measure. The relationship is discussed further below.

Linear utility

The idea of a linear utility measure has been described in previous TREC reports (e.g. [5]). The particular parameters being used are a credit of 2 for a relevant document retrieved and a debit of 1 for a non-relevant document retrieved:

```
Utility = 2*R+ - N+          --> retrieve if P(rel) > .33
```

Filtering according to a utility function is equivalent to filtering by estimated probability of relevance; the corresponding probability threshold is shown.

When evaluation is based on utility, it is difficult to compare performance across topics. Simple averaging of the utility measure gives each retrieved document equal weight, which means that the average scores will be dominated by the topics with large retrieved sets (as in micro-averaging). Furthermore, the utility scale is effectively unbounded below but bounded above; a single very poor query might completely swamp any number of good queries. In TREC-8 we experimented with a range of scaled utility functions for averaging purposes. This year we have taken a simpler approach, which deals crudely with the unboundedness but not with the micro-averaging: a minimum (maximum negative) score is applied. Thus:

```
T9U = Max(2*R+ - N+, MinU)

MinU = -100 for OHSU topics, -400 for MeSH topics
```

Other measures

In the results presented below, and in the official results tables, a number of measures are included as well as the measure for which any particular run was specifically optimised. The range is as follows:

For adaptive and batch filtering:

MnT9P The mean value of the T9P measure over topics

Measures:	T9P:	32 runs
	T9U:	27 runs
	(undeclared):	2 runs

The total of 61 under 'Measures' represents the 61 Adaptive filtering or Batch runs. The two whose optimisation measure was undeclared were evaluated for both measures.

Here is a list of the participating groups, including [abbreviations] and (run identifiers). Participants will generally be referred to by their abbreviations in this paper. The run identifiers can be used to recognize which runs belong to which groups in the plotted results.

- Carnegie-Mellon University, Ault & Yang [CMU-Y] (CMUCAT)
- Carnegie-Mellon University, Zhang & Callan [CMU-C] (CMUDIR)
- Queens College CUNY [CUNY] (pirc)
- Fudan University [Fudan] (FDU)
- Informatique-CDC - Groupe Caisse des Dépôts / ESPCI [ICDC] (S2RN)
- University of Iowa [Iowa] (IOWAF)
- IRIT / University of Toulouse [IRIT] (Mer9)
- Korea Advanced Institute of Science and Technology [KAIST] (KAIST)
- KDD R&D Laboratories [KDD] (kdd)
- Microsoft Research - Cambridge [Microsoft] (ok9)
- University of Montreal [Montreal] (reliefs)
- University of Nijmegen [Nijmegen] (KUN)
- Rutgers University [Rutgers] (ant)
- Seoul [Seoul] (scai)

3.1 Summary of approaches

These very brief summaries are intended only to point readers towards other work. A few of the groups do not have papers in this volume.

Carnegie-Mellon (CMU-Y) have adapted a k-nearest-neighbour text categorization algorithm to the filtering task. In their paper in this volume, they also present arguments against the T9P and T9U measures.

The other Carnegie-Mellon group (CMU-C) used an incremental Rocchio algorithm for query adaptation, and introduced some modifications into this algorithm and their idf term weighting.

CUNY ran last year's adaptive filtering system without modification.

Fudan used a Rocchio-like algorithm, with feature selection using mutual information.

ICDC (routing task) used a neural network without hidden neurons, but with strong feature selection (very few features per topic), with local context.

Iowa used an approach based on dynamic, two-level clustering: each topic has a single first-level cluster, within which further clusters develop.

IRIT's method involved a profile of the non-relevant documents for each profile.

KAIST combined query zoning, a support vector machine and Rocchio's algorithm.

KDD also used a non-relevant document profile, and pseudo-relevance feedback.

Microsoft used limited term selection, and a complex threshold adaptation regime.

Montreal combined the 'document implies query' relevance probability with the reverse implication, and used word conjunctions.

Nijmegen used a method of threshold adaptation based on score distributions, with a Rocchio algorithm. Relevant documents are treated differently according to their date.

Rutgers used Boolean expressions based on the Logical Analysis of Data.

Seoul used a boosted naive bayes method.

3.2 Evaluation results

Some results for the various participating groups are presented in the following tables. Tables 1–4 show the adaptive filtering results on OHSU and MSH-SMP topics, for the two optimisation measures. Various measures are shown in the tables, in addition to the optimisation measure used.

Figures 1–4 show the same result sets broken down by year. In the case of the T9P optimised results, we cannot calculate T9P itself for each year, because the target number of documents applies only to the whole period. Instead, precision is shown. Similarly, it is not appropriate to apply the minimum utility value used in T9U to each individual year – in this case, unadjusted utility is used.

The year graphs for precision might be a little misleading. It would clearly be possible for a system to set its threshold too high at the beginning, so that it obtained good precision but not enough documents; it would then have to relax its threshold in order to retrieve enough documents, but probably get lower precision.

Tables 5–8 show the batch filtering and routing results.

Table 1: Adaptive filtering – OHSU – best T9P results

	MacR	MacP	**MnT9P**	MnT9U	Zeros
Microsoft	38.8	29.4	**29.4**	-6.3	0
CMU-C	41.4	27.9	**27.9**	-5.3	0
Fudan	30.1	27.3	**26.5**	-7.0	0
Nijmegen	29.3	25.8	**25.8**	-7.3	0
CMU-Y	38.0	27.8	**22.4**	-22.1	0
Montreal	17.7	28.0	**16.8**	-1.5	0
Iowa	16.3	19.5	**13.8**	-11.4	5
Rutgers	33.2	15.3	**10.2**	-43.6	7
OHSU topics Runs optimised for T9P Best runs from each group					

3.3 Some comparisons

The new T9P measure provides an interesting opportunity to compare the results of filtering runs with traditional ranked-retrieval runs. The evaluation program trec_eval for ranked retrieval calculates P@n values – precision at n documents retrieved – for various values of n. T9P is a sort of

Table 2: Adaptive filtering – OHSU – best T9U results

	MnT9U	MnSU	MacR	MacP	Zeros
Nijmegen	**17.3**	0.06	23.1	36.7	0
Microsoft	**10.7**	0.01	21.2	33.0	0
CMU-C	**10.1**	0.04	19.0	42.6	0
Fudan	**9.6**	0.00	18.1	31.9	0
Montreal	**1.1**	-0.08	12.0	32.1	1
Iowa	**-5.9**	-0.16	12.9	19.8	6
Rutgers	**-32.3**	-1.75	27.0	14.3	12
KDD	**-35.3**	-0.90	8.3	10.2	0
CUNY	**-55.7**	-11.20	22.2	10.8	0
OHSU topics Runs optimised for T9U Best runs from each group					

Table 3: Adaptive filtering – MSH – best T9P results

	MacR	MacP	**MnT9P**	MnT9U	Zeros
MSH topics					
Microsoft	18.5	41.9	**41.9**	14.8	0
CMU-C	18.5	35.9	**35.9**	19.2	0
Fudan	13.9	35.8	**35.1**	4.5	0
CMU-Y	21.0	35.9	**30.3**	-40.8	105
MSH-SMP topics					
Microsoft	18.9	43.0	**43.0**	16.5	0
CMU-C	18.9	36.3	**36.3**	17.5	0
Fudan	14.2	36.3	**35.6**	5.6	0
CMU-Y	21.4	36.4	**30.4**	-37.4	10
Runs optimised for T9P Best runs from each group					

Table 4: Adaptive filtering – MSH – best T9U results

	MnT9U	MnSU	MacR	MacP	Zeros
MSH topics					
CMU-C	**20.3**	0.07	9.8	47.9	0
MSH-SMP topics					
Microsoft	**46.5**	0.10	18.2	43.6	0
Fudan	**29.3**	0.04	15.4	34.6	0
CMU-C	**26.7**	0.08	13.7	47.1	0
Iowa	**12.9**	0.0	11.5	52.5	52
Runs optimised for T9U Best runs from each group					

Table 5: Batch filtering – OHSU – best T9P results

	MacR	MacP	**MnT9P**	MnT9U	Zeros
Batch-adaptive					
Fudan	37.9	32.2	**31.7**	-1.1	0
Microsoft	38.8	30.5	**30.5**	-5.3	0
Non-adaptive					
CMU-Y	57.4	28.7	**26.1**	-26.9	1
KAIST	22.7	42.1	**20.0**	12.2	0
OHSU topics Runs optimised for T9P Best runs from each group					

Table 6: Batch filtering – OHSU – best T9U results

	MnT9U	MnSU	MacR	MacP	Zeros
Batch-adaptive					
Nijmegen	**19.4**	0.03	41.0	37.6	0
Fudan	**13.6**	0.05	24.5	39.0	0
Non-adaptive					
IRIT	**7.5**	-0.22	21.6	46.5	5
Nijmegen	**5.0**	-0.15	22.6	40.0	3
Seoul	**2.8**	0.02	3.7	80.8	33
OHSU topics Runs optimised for T9U Best runs from each group					

Table 7: Batch filtering – MSH – best T9P & T9U results

	MacR	MacP	**MnT9P**	MnT9U	Zeros
MSH topics – adaptive – T9P					
Fudan	16.1	42.2	**41.8**	14.6	0
MSH topics – non-adaptive – T9P					
CMU-Y	26.0	53.1	**43.6**	4.9	97
MSH-SMP topics – adaptive – T9P					
Fudan	17.6	45.0	**44.0**	19.4	0
Microsoft	18.7	43.3	**43.3**	16.9	0
MSH-SMP topics – non-adaptive – T9P					
CMU-Y	25.7	54.6	**44.3**	11.2	9
KAIST	24.5	54.3	**41.9**	86.4	0
MSH-SMP topics – non-adaptive – T9U					
	MnT9U	MnSU	MacR	MacP	Zeros
Seoul	**20.0**	0.01	5.4	58.8	127
Best runs from each group					

Table 8: Best Routing results

	AveP	P@50	RPrec
OHSU topics			
ICDC	**0.385**	37.0	39.5
Microsoft	**0.326**	33.6	35.1
Nijmegen	**0.237**	28.2	28.6
IRIT	**0.235**	27.9	27.8
Rutgers	**0.182**	21.5	22.9
MSH-SMP topics			
ICDC	**0.335**	53.7	41.0
Microsoft	**0.253**	45.5	32.6
Rutgers	**0.158**	27.0	23.0
Best runs from each group			

P@50[3].

Actually, the comparison between P@50 and T9P is slightly more complex. There are two reasons why we might expect T9P figures to be somewhat lower than P@50:

1. In order to reach a target of 50, we have in fact to aim higher, as discussed above. We would be substantially penalised for not reaching 50. However, going higher is also likely to penalize us somewhat on precision.
2. Even supposing we were to retrieve exactly 50 over the period, they would not necessarily be the best 50 – if we had to adjust the threshold at any stage to achieve the target number at the end, for part of the period we would have retrieved documents in a range of scores which we would have rejected in another part.

As against these arguments, of course, there is the possibility for improving the results through adaptation. In table 9 we show P@50 values for some of the routing runs and T9P for some batch and adaptive filtering runs. The following observations qualify this table: The ICDC and CMU-Y runs made use of the MeSH field of the records. The CMU-Y run was batch non-adaptive; the other two were adaptive. As indicated in the text, the routing and batch filtering runs could make use of all of the relevant documents in the training set. Given that some topics have less than 50 relevant documents, the maximum possible P@50 is 68%.

Although it is clearly possible, for the reasons suggested above, to do better at P@50 than at T9P, it seems that a good adaptive filtering system can overcome much of this handicap.

4 General Commentary

In this section last year, we made the following observation:

> Following the progression of system performance from TREC–7 to TREC–8 (or lack thereof!), it is becoming increasingly clear that the adaptive filtering task is too hard.

We believe, in contrast, that the TREC–9 adaptive filtering task was not too hard, and provided a solid experimental test from which a number of systems emerged with good performances. We may make the following observations in support of this claim:

---

[3] trec_eval does not by default include $n = 50$; however, a simple modification of a header file allows it.

Table 9: Comparing T9P and P@50

Routing: P@50		Batch filtering: T9P		Adaptive filtering: T9P	
ICDC	37.0	Fudan	31.7	Microsoft	29.4
Microsoft	33.6	Microsoft	30.5	CMU-LTI	27.9
Nijmegen	28.2	CMU-Y	26.1	Fudan	26.5
OHSU topics Filtering runs optimised for T9P Best runs from best 3 group					

- In the linear utility version of the task, several systems with non-conservative strategies achieved good positive average utilities.
- With the new precision-oriented measure, several systems achieved effectiveness levels close to those reached in the less-demanding P@50 task in ranked-output retrieval.

It is not immediately clear which particular differences between the TREC–8 and TREC–9 tasks brought about this change. Probably all of the following had some effect, but it would be necessary to do some more diagnostic work to discover their relative importance:

- The use of a small number of positive examples for training;
- The greater number of relevant documents in the test collection;
- Improvements in the systems.

Acknowledgements We give our thanks to all the people who have contributed to the development of the TREC filtering track over the years, in particular David Lewis, Karen Sparck Jones, Chris Buckley, Paul Kantor, Ellen Voorhees, the TREC program committee, and the team at NIST.

References

[1] Hersh, W.R., Buckley, C., Leone, T.J. and Hickam, D.H. OHSUMED: an interactive retrieval evaluation and new large test collection for research. In *SIGIR '94: Proceedings of the 17th Annual ACM SIGIR Conference on Research and Development in Information Retrieval*, Springer-Verlag, pages 192–201, 1994.

[2] I.J. Good. The Decision Theory Approach to the Analysis of Information Retrieval Systems. *Information Storage and Retrieval Systems*, 3:31–34, 1967.

[3] Hull, David A. The TREC-6 Filtering Track: Description and Analysis. In *The 6th Text Retrieval Conference (TREC-6), NIST SP 500-240*, pages 45–68, 1998.

[4] Hull, David A. The TREC-7 Filtering Track: Description and Analysis. In *The 7th Text Retrieval Conference (TREC-7), NIST SP 500-242*, pages 33–56, 1999.

[5] Hull, David A. and Robertson, Stephen The TREC-8 Filtering Track final report. In *The 8th Text Retrieval Conference (TREC-8), NIST SP 500-246*, pages 35–56, 2000.

[6] Lewis, David. The TREC-4 Filtering Track. In *The 4th Text Retrieval Conference (TREC-4), NIST SP 500-236*, pages 165–180, 1996.

[7] David Lewis. The TREC-5 Filtering Track. In *The 5th Text Retrieval Conference (TREC-5), NIST SP 500-238*, pages 75–96, 1997.

TREC-9 CLIR Experiments at MSRCN

Jianfeng Gao[*], Jian-Yun Nie[**], Jian Zhang[#], Endong Xun[*],
Yi Su[#], Ming Zhou[*], Changning Huang[*]

[*] Microsoft Research China, Email: {jfgao, i-edxun, mingzhou, cnhuang} @microsoft.com

[**] Département d'informatique et de recherche opérationnelle, Université de Montréal,
Email: nie@iro.umontreal.ca

[#] Department of Computer Science and Technology of Tsinghua University, China,
Email: ajian@s1000e.cs.tsinghua.edu.cn

Abstract

In TREC-9, we participated in the English-Chinese Cross-Language Information Retrieval (CLIR) track. Our work involved two aspects: finding good methods for Chinese IR, and finding effective translation means between English and Chinese. On Chinese monolingual retrieval, we investigated the use of different entities as indexes, pseudo-relevance feedback, and length normalization, and examined their impact on Chinese IR. On English-Chinese CLIR, our focus was put on finding effective ways for query translation. Our method incorporates three improvements over the simple lexicon-based translation: (1) word/term disambiguation using co-occurrence, (2) phrase detecting and translation using a statistical language model and (3) translation coverage enhancement using a statistical translation model. This method is shown to be as effective as a good MT system.

1. Introduction

In TREC-9, Microsoft Research China (MSRCN), together with Prof. Jian-Yun Nie from University of Montreal, participated for the first time in the English-Chinese Cross-Language Information Retrieval (CLIR) track. Our work involved two aspects: Finding good methods for Chinese IR, and finding effective translation means between English and Chinese.

Finding a good monolingual IR method is a prerequisite for CLIR. On Chinese monolingual retrieval, we examined the problems such as using different entities as indexes, pseudo-relevance feedback, length normalization, as well as cutting documents done into passages. Each of these techniques gave some improvements to Chinese IR. The best combination of them is used for our Chinese monolingual IR.

On English-Chinese CLIR, our focus was put on finding effective ways for query translation. Large English-Chinese bilingual dictionaries are now available. However, beside the problem of completeness of the dictionary, we are also faced with the problem of selecting the best translation word(s) from the dictionary. To deal with this problem, we used an approach called, *improved lexicon-based query term translation*, which bring significant improvements over the simple approach based on bilingual lexicon. In this approach, we investigated the following three problems: (1) word/term disambiguation using co-occurrence, (2) phrase detecting using a statistical language model, and (3) translation coverage enhancement using a statistical translation model.

In section 2, we introduce briefly our work on finding the best indexing unit for Chinese IR. In section 3, we describe in detail the proposed method -- *improved lexicon-based query term translation*, and compare with the method using a machine translation (MT) system in CLIR. In section 4, we describe the use of query expansion techniques. In section 5, experimental results are presented. Finally, we present our conclusion in section 6.

2. Finding the Best Indexing Units for Chinese IR

It is well known that the major difference between Chinese IR and IR in European languages lies in the absence of word boundaries in sentences. Words have been the basic units of indexing in traditional IR. As

---

[**] [#] This work was done while these authors were visiting Microsoft Research China.

BEST COPY AVAILABLE

Chinese sentences are written as continuous character strings, a pre-processing has to be done to segment sentences into shorter units that may be used as indexes. Indexing units for Chinese IR may be of two kinds, words or *n*-grams [Nie, 2000].

When using words, several types of knowledge may be used: manually constructed dictionary that stores a set of known words, heuristic rules on word formation, or some statistical measures based on co-occurrences of characters. A dictionary-based segmentation is widely used to identify all occurrences of the dictionary words in a sentence. If there are word segmentation ambiguities, the longest-matching strategy is usually used to select the best choice. There are mainly two problems of this approach. The first is the loss in recall. A long word may contain several shorter words. In the longest matching, only the longest word is identified as an index, and all the included short words are ignored. For example, if 操作系统 (operating system) is identified as a word, 操作 (operating) and 系统 (system) will not. However, in practice, we also refer to an "operating system" by just "system". Although the word "system" is included in "operating system", it will not be considered as a completely independent index for IR. Therefore some relevant documents will not be retrieved. The second problem is the unknown word problem. Especially, many proper nouns, which play an important role in IR, are not in the dictionary, and are not considered as indexes.

Another kind of indexing units is *n*-grams. This method does not require any linguistic knowledge. Usually, one chooses n-grams of lengths 1 or 2 (uni-grams or bi-grams). Longer *n*-grams are rarely used due to the higher memory cost and their marginal improvement over bi-grams. In comparison with words, the advantage of bi-grams lies in its robustness to unknown words. For example, for proper nouns that are not in the dictionary, such as 大亚湾 (a place in southern China), word segmentation will segment the proper noun into three characters, i.e. 大, 亚, and 湾. When using bi-grams, we can still use part of the proper nouns as indexes, i.e. 大亚, 亚湾. If both bi-grams occur in the same document, there is a higher probability that the document concerns 大亚湾, than the documents where the three single characters occur. Political terms or abbreviations (e.g. 三乱 – three turmoils), and foreign names (e.g. 皮纳图博火山 - Mount Minatubo) are similar examples showing the advantage of using bi-grams.

Words and n-grams represent two different ways to represent a document – one relies on linguistic knowledge and the other on statistical information only. It is a common practice to combine different evidence to judge document relevance. So it is also reasonable to combine *n*-grams with words.

To sum up, we can create three possible representations for a document and a query as shown in figure 1, i.e. words, characters, and bi-grams. We also see that some correspondences may be created across representations, if different representations are integrated (for example, between words and characters).

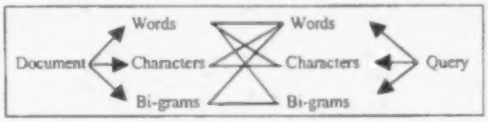


Fig. 1. Possible representations in Chinese IR

In order to determine the best indexing units, we conduct a series of test tests on TREC 5&6 Chinese data [Harman, 1996]. The documents in the collection are articles published in the People's Daily from 1991 to 1993, and a part of the news released by the Xinhua News Agency in 1994 and 1995. A set of 54 English queries (with translated Chinese queries) has been set up and evaluated by people in the NIST (National Institute of Standards and Technology).

Once Chinese sentences have been segmented into separate items, traditional IR systems may be used to index them. These separate items are called "terms" in IR. For our experiments, we used a modified version (the modifications are made in order to deal with Chinese) of the SMART system [Buckley, 1985].

The following methods have been compared:

1. using the longest matching with a small dictionary and with a large dictionary
2. combining the first method with characters
3. using full segmentation with or without adding characters
4. using bi-grams and characters
5. combining words with bi-grams and characters

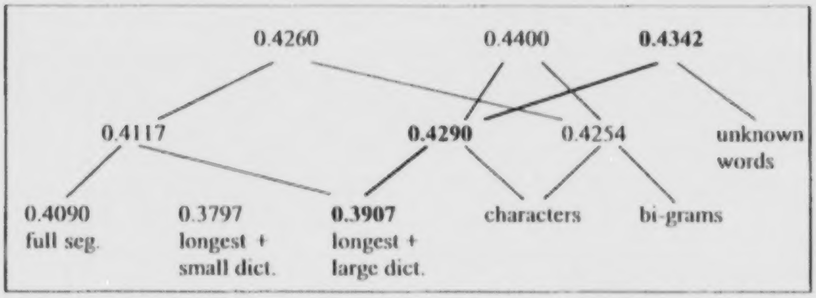


Fig. 2. Results of indexing units for Chinese IR

The results of this series of experiments are summarized in the figure 2, detailed description can be found in [Nie, 2000].

In order to examine the impact of dictionary in word segmentation, two different dictionaries are used. The small dictionary contains 65,502 entries. The large dictionary contains 220K entries, containing not only all entries in the small dictionary, but also a large number of phrase, including date expressions (e.g. 一九三四年 - year 1934), suffix structures (e.g. 使用者 - user), etc. The second dictionary is more complete. In both cases, we use the same forward longest-matching strategy. Using the first dictionary, we obtained an average precision of 0.3797. Using the second dictionary, the average precision is increased to 0.3907. We can see that a better dictionary can increase IR effectiveness to some extent.

To remedy the loss in recall caused by the use of the longest words, we complement the longest words by single characters. We obtain nontrivial improvements. In the case of large dictionary, we achieve an average precision of 0.4290 (9.8% improvement). It turns out that simply adding single characters is a more effective way to increase IR performance than increase the dictionary size. Another way to increase recall is to extract the short words implied in long words, called full segmentation. In this case, we obtain an average precision of 0.4090. Although the performance is better than using the longest words alone, it is worse than the method by adding single characters. One of the reasons might be due to the cross-word segmentation phenomenon; i.e. some words extracted in full segmentation are composed of parts of two different words. For example, from the string 开发油田 (exploit a oilfield), we not only extract the correct words 开发 (exploit) and 油田 (oilfield), but also 发油 (hair oil).

Previous studies [Kwok, 1997; Nie, 1999] show that when combining bi-grams with uni-grams, the IR performance is better. We repeat this experiment here, and obtained an average precision of 0.4254. This performance is comparable to the best performance we obtained using words. This is largely contributed to the robustness for dealing with unknown words by *n*-grams.

As bi-grams and words have their own advantages, we try to combine them to benefit from both. Theoretically, such a combination would result in a better precision (due to words) and an increased robustness for unknown words (due to n-grams). Unfortunately, the experimental result is not promising enough. We obtain slightly improvements of 2.6% (average precision 44.00%) over the uncombined case, whereas the space and the time of indexing are more than doubled.

After word segmentation, we noticed that some important proper nouns and noun phrases have not been recognized as words, but segmented into single characters, such as 皮纳图博火山 (Mount Minatubo). Therefore, we used NLPWin[1] to recognize multi-word phrases and unknown words. NLPWin first tags texts using a Chart-parser (with a dictionary). For unknown words, a category is guessed according to its context. Special rules have also been integrated to recognize proper nouns. As a consequence, most Chinese or English proper nouns can be tagged and recognized correctly. Some political terms and abbreviations (e.g. 中越 - Sino-Vietnam) can also be recognized. Using NLPWin, we created another set of words that is added

---

[1] The NLPWin system is a natural language processing system developed by Microsoft Research. The system converts text into a parse tree that represents the syntactic structure and then into its logical form that reflects the meaning of the text. These representations can then be used for tasks such as grammar checking, machine translation, and information retrieval.

	riginal vg.P.	ew vg.P.	Impr.	New words added
9	0.3648	0.4173	14.4%	毒品买卖 (drug sale)
23	0.3940	0.5154	30.8%	联合国安理会 (Security committee of UN), 和平建议 (peace proposal)
28	0.4824	0.5034	4.4%	蜂窝式 (cellular), 交换网 (interchange network)
46	0.3483	0.4192	20.4%	中越 (Sino-Vietnam)
47	0.5369	0.5847	8.9%	皮纳图博火山 (Mount Minatubo), 臭氧层 (ozone layer)
54	0.6778	0.7005	3.3%	F-16, 八. 一七 (August 17)

Table 1: Impact of unknown word recognition on some queries.

to our original dictionary. From the 54 queries, 80 new words have been recognized. Most of them are proper nouns or noun phrases. The addition of unknown words had positive impact for 10 queries out of 54, while the effectiveness is reduced for 4 queries. Table 1 contains some examples of queries for which the addition of new words has positive impacts. As we can see in Fig. 2, the global effect of adding an unknown word detection is positive.

We can see from figure 2 that as long as different kinds of indexes are combined the IR performance increases. The question now is whether the combination is worth the cost. In taking into account both effectiveness and cost, we think the combination should go in the direction represented by the bold lines in figure 2. For our experiments in TREC9, we will use the combination of the longest words, single characters and detected unknown words for Chinese IR.

3. Query Translation

The methods for query translation, proposed recently, fall into three categories: (1) using MT systems, (2) using parallel corpora, and (3) using bilingual lexicons. The third method is the simplest way to implement because of its simplicity and the increasing availability of machine-readable bilingual lexicons. Therefore, we decided to start with this method in TREC9 and try to improve it by adding other tools.

The main problems we observe on this simple method are: 1) the dictionary used may be incomplete; and (2) it is difficult to identify the correct word sense from the lexicon. To deal with these issues, we used an ***improved lexicon-based query translation***. It tries to improve the lexicon-based translation through (1) word/term disambiguation using co-occurrence, (2) phrase detecting and translation using a statistical language model, and (3) translation coverage enhancement using a statistical translation model. In what follows, we will describe each of them in detail.

3.1 Word/term disambiguation

It is assumed that the correct translations of query terms tend to co-occur in target language documents and incorrect translations do not. Therefore, given a set of original English query terms, we select for each term the best translation term such that it co-occurs most often with other translation words in Chinese documents. Finding such an optimal set is computationally very costly. Therefore, an approximate algorithm is used. It works as follows. Given a set of n original query terms $\{s_1,\ldots,s_n\}$, we first determine a set T_i of translation words for each s_i through the lexicon. Then we try to select the word in each T_i that has the highest degree of cohesion with the other sets of translation words. The set of best words from each translation set forms our query translation.

The cohesion is based on term similarity calculated as follows. For terms x and y, their similarity is:

$$SIM(x,y) = p(x,y)\times\log_2\left(\frac{p(x,y)}{p(x)\times p(y)}\right) - K\times\log_2 Dis(x,y) \qquad (1)$$

where

$$p(x,y) = \frac{c(x,y)}{c(x)} + \frac{c(x,y)}{c(y)}$$

$$p(x) = \frac{c(x)}{\sum_{x} c(x)}$$

and *c(x,y)* is the frequency that term x and term y co-occur in the same sentences in the collection, *c(x)* is the number of occurrence of term x in the collection, *Dis*(*x*,*y*) is the average distance (word count) between term x and term y in a sentence, and K is a constant coefficient.

The cohesion of a term x with a set T of other terms is the maximal similarity of this term with every term in the set, i.e.

$$Cohesion\ (x, T) = Max_{y \in T}\, SIM\ (x, y)$$

The greedy algorithm used to select the word translations is as shown in figure 2.

The term-similarity matrix is obtained via a statistical model, which is trained using a large Chinese corpus of MSRCN consisting of 1.6 billion characters.

```
For each $s_i$ ($i$ = 1 to $n$), retrieve a set of senses $T_i$ from the lexicon;
  For each set $T_i$ ($i$ = 1 to $n$), do
      For each term $t_{ij}$ in $T_i$, do
    For each set $T_k$ ($k$ = 1 to $n$ & $k$<>$i$), compute the cohesion $Cohesion(t_{ij}, S_k)$;
Compute the score of $t_{ij}$ as the sum of $Cohesion(t_{ij}, S_k)$ ($k$ = 1 to $n$ & $k$<>$i$);
Select the term $t_{ij}$ in $T_i$ with the highest score, and add the selected sense into the set $T$.
```

Fig. 3. Greedy algorithm to find best translations

3.2 Phrase detecting and translation

The translation of multi-word phrases is usually more precise than a word-by-word translation [Ballesteros, 1998], since phrases usually have fewer senses. However, if a phrase is not stored in a lexicon, we usually can do nothing. Unfortunately, in TREC-9 query set, more than 50% phrases are not in our lexicon.

In our experiments, we try to incorporate some translation patterns between English and Chinese. For example, a (NOUN-1 NOUN-2) phrase is usually translated into the (NOUN-1 NOUN-2) sequence in Chinese, and a (NOUN-1 of NOUN-2) phrase is usually translated into the (NOUN-2 NOUN-1) sequence in Chinese. So if we can detect the English phrase of some patterns, we can guess the form(s) of the translation phrases. For instance, the translation of the multi-word phrase "drug sale" is 毒品(drug)/买卖(sale), and the translation of the multi-word phrase "security committee of UN" is 联合国(UN)/安理会(security committee).

To do this, we use again NLPWin to detect phrases in the English queries. We selected a set of 40 English patterns (PAT_{Te}) that are often used in phrases. For each of them, we estimate the probability of the order of translation words, $p(O_{Tc}|PAT_{Te})$. Then the best translation phrase is the one that maximizes the following function,

$$Tc = \text{argmax}\ p(O_{Tc}|PAT_{Te})\ p(Tc) \qquad (2)$$

where $p(Tc)$ is a priori probability whose value is given by the bigram language model. The bigram language model is trained using the same large Chinese corpus of MSRCN. For the moment, an approximate probability $p(O_{Tc}|PAT_{Te})$ is assigned by a linguist because of the lack of training data.

3.3 Using translation model

Translations stored in lexicons are always limited, no matter how complete they are. Parallel texts may contain additional translations. Therefore, we used a statistical translation model trained from a set of parallel texts as a complement of the previous methods.

Given a set of parallel texts in two languages, they are first aligned into parallel sentences. While the lexically based techniques use extensive online bilingual lexicons to match sentences [Chen 93], statistical techniques require almost no prior knowledge and are based solely on the lengths of sentences, i.e. length-based alignment method. We use a novel method that incorporates both approaches [Liu, 95]. First, the rough result is obtained by using the length-based method. Then anchors are identified in the text to reduce the complexity. An anchor is defined as a block that consists of $n=3$ successive sentences. Finally, a small, restricted set of lexical cues is applied to obtain further improvements.

Once a set of parallel sentences is obtained, word translation relations are estimated. Chinese sentences are first segmented into word strings by using a dictionary, containing approximately 80 thousand words, in conjunction with an optimization procedure described in [Gao, 2000]. The bilingual training process employs a variant of the model in [Brown, 1993] and it is based on an iterative EM (expectation-maximization) procedure for maximizing the likelihood of generating the English given the Chinese portion. The output of the training process is a set of potential Chinese translations for each English word, together with the probability estimate for each translation.

The problem we often have with translation models is the unavailability of parallel texts for Chinese-English. To solve the problem, we conducted a text-mining project in the Web to find parallel texts automatically [Nie, 1999]. We select about 20,000 parallel document URLs, from which 870,414 pairs of sentences are selected for model training. The training data amounts to 74MB Chinese texts and 51MB English texts.

Let's assume that all multi-word phrases have been translated by equation (2). By combining translation model, we can arrive at the following equation of query phrase translation:

$$Tc = \arg\max \; p(Te \mid Tc)\, SIM\,(Tc) \qquad (3)$$

where $p(Te|Tc)$ is the translation probability of Chinese term Tc to English term Te, and SIM(Tc) is the sum of the maximum similarity score of the selected translation set Tc, which is estimated by algorithm in figure 2 and equation (1).

3.4 Tests of Query Translation on TREC 5&6

We carried out a series of tests to compare our improved method with the following four cases:

1. *monolingual*: retrieval using provided (manually translated) Chinese queries;
2. *simple translation*: retrieval using query translation obtained by looking up the bilingual lexicon;
3. *best-sense translation*: retrieval using query translation obtained by manually selecting the best senses among the senses in the bilingual lexicon for each query term;
4. *machine translation*: retrieval using translation queries obtained by the machine translation software system.

In our experiments, the English-Chinese bilingual lexicon we used comes from LDC (http://morph.ldc.upenn.edu/Projects/Chinese/). It contains 110,834 English entries as well as their corresponding Chinese translations. For each English entry, there are usually several Chinese translations. The *simple translation* works in two modes. One is u-mode that selects the most Chinese translation for each English term. The other is m-mode that selects the first three (if it contains no less than three translations) frequent-used Chinese translations.

For *best-sense translation*, we manually select one translation for each term in queries, for multi-word phrases not found in the lexicon, we translate it word-by word.

The *improved translation* makes use of the following tools described previously: (1) the term-similarity matrix for term disambiguation, (2) the language model for phrase translation, and (3) the translation model for lexicon coverage enhancement.

The use of an MT package is convenient for CLIR since it takes care of problems like word morphology, parsing, etc. On the other hand, its internal working scheme and dictionaries are proprietary, and one can only treat it as a black box and has to accept the output as is with little possibility of changing them. In our experiments, a commercial English to Chinese machine translation software system called IBM HomePage Dictionary™ 2000 is used. The system is released recently by IBM. It contains a 480K English-Chinese dictionary, which consists of both words, frequently used phrases (such as "information retrieval"), acronyms (such as "IBM"), and proper nouns (such as "Microsoft"). It can translate a word, phrase, sentence or whole document. According to our survey, this system is one of best machine translation product currently on the market. The result of query translation by the IBM system seems reasonable; less than 2% of the words are left untranslated, most phrases are translated as a whole, and the ambiguity problem of most words are solved successfully.

The results of this series of experiments on query translation are summarized in table 2. As can be expected, the simple translation methods are not very good. Their performances are lower than 60% of the monolingual performance.

The best-sense method improves the performance of the simple translation method. It achieves 73.05% of monolingual effectiveness. However, it is still worse than our improved translation method, which achieves a 75.40% performance of that of monolingual IR.

IBM HomePage Dictionary™ 2000 is a very powerful machine translation system. Using it for query translation, we can achieve 75.55% of monolingual effectiveness. On the other hand, the fact that the most powerful commercial machine translation system performs almost the same as our improved method indicates the effectiveness of our query translation technique for CLIR.

The best performance is achieved by combining linearly two sets of translation queries obtained by machine translation method and the improved translation method. It is over 85% of monolingual effectiveness. The motivation of combination of different translation methods is that different translation systems would complement each other.

	Translation Method	Avg.P.	% of Mono. IR
1	Monolingual	0.5150	*
2	Simple translation(m-mode)	0.2722	52.85%
3	Simple translation(u-mode)	0.3041	59.05%
4	Best-sense translation	0.3762	73.05%
5	Improved translation	0.3883	75.55%
6	Machine translation	0.3891	75.40%
7	5 + 6	0.4400	85.44%

Table 2: Average retrieval precision of the English translation queries.

4. Query Expansion

4.1 Pre-translation & Post-translation Query Expansion

Earlier work showed that query expansion can greatly reduce the error associated with dictionary translation [Ballesteros, 1998]. A popular method of query expansion in TREC experiments is the 2-stage pseudo relevant feedback. At first, raw queries are used to retrieve a ranked list of documents. Then the set of *n* top-ranked documents is used for query expansion. Usually, we expand the initial query by adding *m* top-frequent terms from the *n* top-ranked documents. Through a preliminary experiment, we established the optimal values (with respect to our test collection) of *n* and *m*.

In CLIR, queries can be expanded prior to translation, after translation or both before and after translation. In English-Chinese CLIR, pre-translation query expansion means using a separate English collection for pre-translation retrieval in order to expand the English query with highly associated English terms. These terms may help focus on the query topic and bring more translated terms that together are useful for disambiguating the translation.

4.2 Sub-Documents[2]

The purpose of dividing a document into a sequence of subdocuments (or passages) of certain length is to create a length normalization effect. It is also hoped that each passage will concentrate on a specific topic, or at least on fewer topics than a complete document. Real documents can be very long (e.g. 2 MB) and very short (e.g. a few words). When such documents form the top-ranked pool, one would face a lot of noise during term selection. Using sub-documents have the advantage of being able to define a more specific domain that is less noisy for query expansion. In our experiments, the medium length of subdocument is set at 550 words. We used pivot normalization [Singhal, 1996] in Smart (the *ltu* weight scheme), given the old weight, w, of a term, the new weight, w', can be written as:

$$w' = \frac{w}{(1.0 - slope) \times pivot + slope * nbTerm} \quad (4)$$

where *pivot* is the average numbers of terms in a documents, *nbTerm* is the actual number of terms in the current document, *slope* is a parameter determining the impact of document length normalization, and a typical setting is *slope* = 0.1.

4.3 Tests of Query Expansion on TREC 5&6

We conducted another series of experiments to measure the effectiveness of our query expansion techniques. The experimental results on monolingual IR are shown in table 4. The indexing units used in this case are the longest words and single characters. The query expansion was performed by adding the top 500 terms from the top 20 documents of the initial ranked documents. When using the SMART *ltc* weighting scheme, we obtained 9.1% improvement over the initial retrieval. Move improvements are obtained when we do retrieval and feedback using sub-documents of a certain size (550 words). The document length normalization, i.e. *ltu*, also leads to limited improvements. It is interesting to note that the best result is achieved when we use the *ltc* weighting scheme at the 2-stage retrieval, but keep the *ltu* at the 1-stage retrieval (last row of the table 3).

---

[2] The idea of sub-document and its implementation details are introduced by Prof. K.L. Kwok during his one-month visit at MSRCN in June, 2000.

Sub-doc	1-stage – weighting	2-stage – weighting	1-stage: Avg.P.	2-stage: Avg.P.
No	*ltc*	*Ltc*	0.429	0.476
Yes	*ltc*	*Ltc*	0.435	0.485
Yes	*ltu*	*Ltu*	0.461	0.489
Yes	*ltu*	*Ltc*	0.461	0.515

Table 3: Average retrieval precision of the expanded queries for Chinese IR.

Method	Avg.P.	% 1-stage
1-stage retrieval	0.3249	*
1+Post-translationQE	0.4280	31.7%
2+Pre-translation QE	0.4400	35.4%

Table 4: Average retrieval precision of the expanded queries.

The overall results of query expansion on CLIR are shown in table 4, which provides the average retrieval precision of 1-stage retrieval (without query expansion) as a baseline, as shown in row 2.

Post-translation expansion was performed by adding the top 500 terms from the top 20 documents of the initial ranked documents after query translation. It brings about 31.4% of improvements, as shown in row 3,

We experimented with pre-translation query expansion using the Foreign Broadcasting collections of TREC and used various levels of query expansion. An English query is first used to retrieve a set of documents from this collection. The top 10 English terms from the top 10 documents are used for query expansion before query translation. As shown in Table 4, the pre-translation QE brings an additional improvement of about 2.8% compared to not using it.

5. Experiments in TREC 9

5.1 Data

The documents in the TREC 9 Chinese collection are articles published in Hong Kong Commercial Daily, Hong Kong Daily News, and Takungpao. Some statistical data are shown in table 5. A set of 25 English queries (with translated Chinese queries) has been set up and evaluated by people in the NIST.

Source	Dates	Size
Hong Kong Commercial Daily	8/98-7/99	~100MB
Hong Kong Daily News	2/99-7/99	~80MB
Takungpao	9/98-9/99	~80MB

Table 5: TREC 9 data.

5.2 Results

We submitted 5 runs, as shown in Table 6.

The monolingual run (MSRCN1) uses the longest words, single characters as well as automatically detected unknown words for indexing. The weighting scheme used is that *ltu* is used for the 1-stage retrieval and *ltc* for the 2-stage retrieval.

The MSRCN2 run is the one in which our improved method is combined with the IBM MT system. No pre-translation QE is used.

Run #	Avg.P.	% of mono. IR	Method
MSRCN1	0.2995	*	Mono-lingual IR
MSRCN2	0.3083	102.9%	CLIR without pre-translation query expansion
MSRCN3	0.2974	99.3%	CLIR with pre-translation query expansion
MSRCN4	0.2677	89.4%	CLIR with *improved translation* only.
MSRCN5	0.2623	87.6%	CLIR with IBM MT system only

Table 6: Average precision of the submitted runs

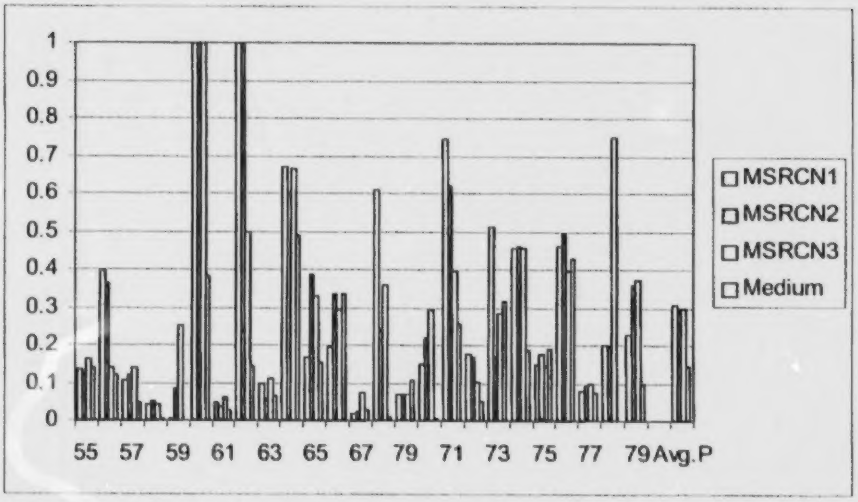


Fig. 4. TREC-9 results for 25 queries

Method	≥ Medium	< Medium
MSRCN1	20	5
MSRCN2	19	6
MSRCN3	20	5

Table 7. Comparison with medium

MSRCN3 run uses the same combination, but with a pre-translation QE.

MSRCN4 and MSRCN5 use respectively our improved method and the IBM MT system alone. Both pre-translation QE and post-translation QE are used in both cases.

As indicated in table 5, unlike the experimental results on TREC5&6, pre-translation QE does not obtain any improvements. The similar effectiveness of MSRCN4 and MSRCN5 shows again that our approach leads to almost the same effectiveness as the IBM MT system. It is also interesting to find that the best CLIR performance is over 100% of the monolingual. In order to analyze how good our query translation approach for CLIR, we display in Fig. 4 a comparison of the retrieval results for the 25 queries. Another comparison with the medium performance is given in Table 7.

Through our first analysis, the queries may be classified into three categories:

1) 5 queries that have both monolingual and CLIR result of Avg.P lower than 0.1. They are #58, 61, 67, 69, and 77. The bad effectiveness in these cases is not due to translation, but because the query topics are difficult for IR.

2) 11 queries with monolingual Avg.P lower than CLIR. There might be two possible reasons. The first is due to the multiple translations for some key words by combining different translation methods, i.e. our approach and IBM MT software. These multiple translations usually are exchangeable. Multiply translations act as the query expansion. Some examples are: "public key" in query 68# is translated to "公共密钥" as well as "公共密码", "Olympics" in query 70# to "奥林匹克" (Olympic) and "奥运会" (Olympic games), and "Panda bear" in query 76# to "大熊猫" and "大猫熊", etc. The second reason is due to better translations over the original ones. For example, "violation" in query #56 is translated to the more common "侵害" rather than "违反".

3) 9 queries with monolingual Avg.P higher than CLIR. Most of them are due to the bad translations of key concepts. For example, query 65# contains an important term "three-links" (三通), a political abbreviation. This term is not translated correctly. This situation is very similar to some cases observed in TREC5&6, where we encountered the terms such as "most-favor nation" (最惠国), "World Conference on Women" (世妇会), and "Project Hope" (希望工程).

Some domain specific composition phrases, which are not included in the lexicon, such as "stealth technology" (隐秘技术) and "stealth countermeasure" (反隐秘技术) in #59, "computer hacker" (电脑黑客) in #65, "synthetic aperture radar" (合成孔径雷达) in #66, "vehicle fatalities" (车祸) in #68 have special terminology in Chinese and are also not picked up, although every word in each phrase is given the correct sense. Other cases are due to the wrong translations of words, for example, "livestock" in #69 is translated to "牲畜", but the correct translation in this query should be "畜牧业", which is not included in the lexicon.

6. Conclusion

In this paper, we described our work in the TREC-9 evaluation in the English-Chinese Cross-Language Information Retrieval (CLIR) track. It involved two aspects: finding good methods for Chinese IR, and finding effective translation means between English and Chinese.

On Chinese monolingual retrieval, we examined the problems such as using different entities as indexes, pseudo-relevance feedback, length normalization, as well as cutting documents done into passages. Each of these techniques gave some improvements to Chinese IR. The best combination of them is used for our Chinese monolingual IR.

On English-Chinese CLIR, our focus was put on finding effective ways for query translation. We have a large English-Chinese bilingual dictionary from LDC. However, beside the problem of completeness of the dictionary, we are also faced with the problem of selecting the best translation word(s) from the dictionary. To address this problem, the following complementary tools have been used: (1) word/term disambiguation using co-occurrence, (2) phrase detecting and translation using language model, and (3) translation coverage enhancement using translation model.

The experimental results we obtained are very encouraging. On Chinese monolingual IR, we obtained 51.50% for TREC5 and 6 Chinese data. This is favorably comparable to the best effectiveness achieved in the previous Chinese TREC experiments.

On English-Chinese CLIR of TREC5 and TREC6, we obtained 75.55% of monolingual effectiveness using our approach. To compare with an MT system, we also tested the IBM MT system, which, when used alone, leads to the same effectiveness (75.40%). When our approach is combined with IBM MT system, we obtained over 85% of monolingual effectiveness. This shows that some translation tools specially designed for query translation may be as suitable as a high-cost MT system, and even if a high-quality MT system is available, our approach can still lead to additional improvements.

Acknowledgement.

The authors would like to thank Prof. K.L. Kwok for his helpful suggestions, and Aitao Chen for his comments on the paper.

Reference

[Ballesteros, 1998] L. Ballesteros, and W. B. Croft. Resolving ambiguity for cross-language retrieval. In *Proceedings of the 21st International Conference on Research and Development in Information Retrieval*. Melbourne, Australia, 1998.

[Brown, 1993] Brown, P. F., S. A. Della Pietra, V.J. Della Pietra, and R.L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2): 263-311

[Buckley, 1985] Buckley, C. *Implementation of the SMART information retrieval system*, Technical report, #85-686, Cornell University, 1985.

[Chen 93] Chen, Stanley F.(1993). Aligning sentences in bilingual corpora using lexical information. In *Proceedings of the 31st Annual Conference of the Association for Computational Linguistics, 9-16, Columbus, OH.*

[Gao, 2000] Jianfeng Gao, Han-Feng Wang, Mingjing Li, and Kai-Fu Lee, 2000. A Unified Approach to Statistical Language Modeling for Chinese. In *IEEE, ICASPP2000*.

[Harman, 1996] Harman, D. K. and Voorhees, E. M., Eds. *Information Technology: The Fifth Text REtrieval Conference (TREC-5)*, NIST SP 500-238. Gaithersburg, National Institute of Standards and Technology, 1996.

[Kowk, 1999] K.L. Kowk, English-Chinese cross-language retrieval based on a translation package. In *Proceedings of the 22nd International Conference on Research and Development in Information Retrieval*. 1999.

[Kwok, 1997] Kwok, K. L. Comparing representations in Chinese information retrieval. *Conference on*

Research and Development in Information Retrieval, ACM-SIGIR, 1997, pp. 34-41.

[Liu, 95] Xin Liu, Ming Zhou, Shenghuo Zhu, Changning Huang (1998), Aligning sentences in parallel corpora using self-extracted lexical information, *Chinese Journal of Computers (in Chinese), 1998, Vol. 21 (Supplement): 151-158.*

[Nie, 1999] Nie, J.-Y., Ren, F. Chinese information retrieval: using characters or words? *Information Processing and Management*, 1999, 35: 443-462.

[Nie, 2000] Jian-Yun Nie, Jianfeng Gao, Jian Zhang, and Ming Zhou. On the use of words and n-grams for Chinese information retrieval. In the *Fifth International Workshop on Information Retrieval with Asian Languages, IRAL-2000.* Hong Kong, September 30 to October 1, 2000.

[Salton, 1983] Gerard Salton and M. J. McGill. Introduction to modern information retrieval. McGraw Hill Book Co., New York, 1983.

[Singhal, 1996] Amit Singhal, Chris Buckley and Mandar Mitra. Pivoted document length normalization. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, 1996, Pages 21 – 29.

[Zhang, 2000] Jian Zhang, Jianfeng Gao, Ming Zhou. Extraction of Chinese compound words – an experimental study on a very large corpus. The second Chinese Language Processing Workshop, Hong Kong, October 8, 2000.

Question Answering using a large NLP System

David Elworthy
Microsoft Research Limited, St. George House, 1 Guildhall Street, Cambridge CB2 3NH, UK

1 Introduction

There is a separate report in this volume on the Microsoft Research Cambridge participation in the Filtering and Query tracks (Robertson and Walker 2001).

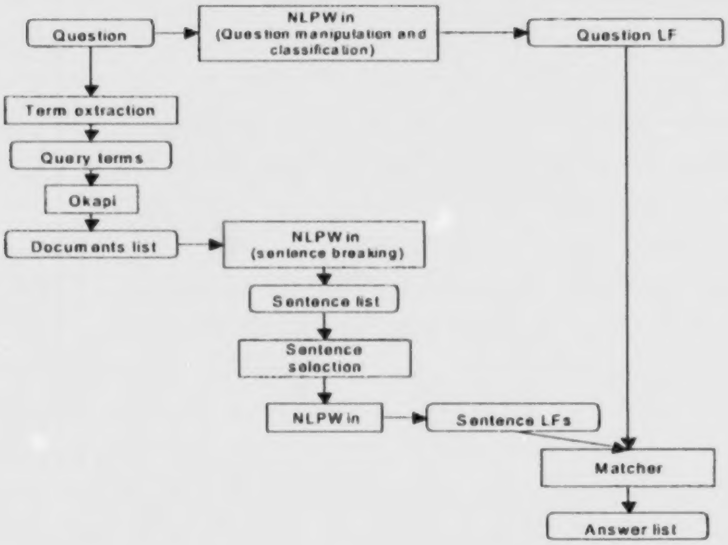
The Microsoft Research question-answering system for TREC-9 was based on a combination of the Okapi retrieval engine, Microsoft's natural language processing system (NLPWin), and a module for matching logical forms. There is no recent published account of NLPWin, although a description of its predecessor can be found in Jensen et al. (1993). NLPWin accepts sentences and delivers a detailed syntactic analysis, together with a logical form (LF) representing an abstraction of the meaning. The original goal was to construct a framework for complex inferencing between the logical forms for questions and sentences from documents. Many answers can be found with trivial inference schemas. For example, the TREC-8 question *What is the brightest star visible from Earth?* could be answered from a sentence containing ... *Sirius, the brightest star visible from Earth* ...by noting that all of the content words from the question are matched, and stand in the same relationships in the question and in the answer, and that the term *Sirius* is equivalent to the answer's counterpart of the head term in the question, *star*. The goal of using inferencing over logical forms was to allow for more complex cases, as in *Who wrote the play ``Hamlet"?* which should not be answered using ... *Zeferelli's film of ``Hamlet"* since a film is not a play. The idea of using inferencing for question-answering is not new. It can be found in systems from the 1970s for story understanding (Lehnert, 1978) and database querying (Bolc, 1980), and in more recent work for questions over computer system documentation (Aliod, 1998).

Time pressure forced this idea to be dropped (work on the system did not start until March 2000), and instead a simpler scheme was adopted, still using LFs from NLPWin. The main observation behind the actual system is that the answer often appears in close proximity to the content terms from the question within the LF, as in the Sirius example above. Consequently, we can try to find answers by identifying candidate nodes in the LF and then using a measure of the proximity. For some kinds of question, such as *when* questions, there is a clear way of identifying candidate answers; for others, such as *what*, it is much harder.

In the following section, we will look at the architecture of the system, and describe the main question types and how they are handled. The evaluation follows in section 3. The results turned out to be relatively poor. Interestingly, there is a very large difference between the results on the TREC-8 test set and the TREC-9 questions, and we will use a fine grained evaluation to examine why.

2 Method

The architecture of the system is shown below. The question is analysed by NLPWin to produce a logical form, and in addition a set of query terms is extracted from it. The query terms will normally contain all of the words of the question less the question word itself (*what*, *who*, *how*, etc.) and a few other stop words. The query terms are used by the Okapi IR engine with BM25 weighting to produce a list of documents. The documents are then segmented into sentences. This stage uses NLPWin, although without using its detailed linguistic analysis capabilities. The resulting list of sentences is ordered by the number of terms from the question they contained, and processed again by NLPWin, this time producing the full linguistic analysis. A cutoff on the number of sentences is used to control the processing time, since a full NLP analysis can be quite time-consuming. The resulting logical forms are compared with the question's logical form to produce a ranked list of answers with scores.



An example of a logical form appears below, for the question *What is the brightest star visible from Earth?*

```
be1 (+Pres +WhQ +L1)
|_Dsub---star1 (+Def +Pers3 +Sing +Conc +Count)
         |_Attrib+-bright1 (+Supr +PosSupr +A0)
                 +-visible1 (+PostNom +E0)
                   |_from---Earth1 (+Pers3 +Sing +PrprN +Conc +Count +Mass)
|_Dnom---what1 (+Wh +Pers3 +Sing)
```

The nodes of the graph (in bold) generally represent the content terms of the analysed sentence, although a few nodes (such as *bel* and *what1*) are more of a structural nature. The nodes are connected by directed relations such as *Dsub* and *Attrib*. Each node can have a number of binary properties, such as *+WhQ*. What we show here is a simplified version of the LF, and the full internal representation contains more information. There is a very large number of different relation types and properties, and we will not attempt to list them here.

2.1 Question manipulation and classification

The aim of the question manipulation stage is to simplify the logical form of questions in order to make it easier to classify them, and to label certain terms in the question as formal and hence not expected to match a term in a candidate answer.

The majority of the manipulations look for a specific question word, attached to a specific relation. For example, a question of the form *Who is X* receives a logical form in which *X* has an *Equiv* relation to a node for *who*. In such cases, we simply delete the relation and who and add an annotation to the top node of *X* which indicates that we are looking for an answer to a *Who* question over objects with the property of being *X*. Similar principles apply to many of the question types. The relation may be other than *Equiv*; for example in *where* and *when* questions, the relations *Locn* and *Time* are used. A second case which occurs frequently is logical forms in which the topmost node is *be*, usually with a single child, or with one child which is a Wh-word and one which is a content node. In such cases, we remove the *be* node, and in the latter case move the Wh-word's properties to the other child.

There are some common subjects for *what* questions, such as *what country...*, *what year....* In these cases, we remove the whole what-phrase and mark the remaining top level node with a special property to indicate that the question should be answered with a restriction as to the answer type. This is only done when the subject corresponds to a property which NLPWin marks in the LF, such as *Cntry* for country. NLPWin derives this information from its lexical resources.

Tthere are a number of other question manipulations on broadly similar principles. After the manipulation, we then assign each question to a category, using the question word (often now discarded and encoded as a property) and the structural configuration. An example of a distinction made using the structure comes with *who* questions, where we distinguish questions asking about identity, as in ***Who is the leader of India?*** from questions about a role of a predicate, as in ***Who invented the paper clip?***. A full list of the question types appears in the appendix. A few questions are left as having *Unknown* type, and questions with an incomplete parse are assigned the type *Bad*.

2.2 Matching

Matching proceeds by selecting and scoring possible answers guided by the question type, and then by extracting the phrase to return as the result. Answer selection is the most complex part of the matching process, and we return to it in a moment. The result of answer selection is a node in the logical form of the answer sentence and a score. To extract the answer, we look up the syntactic node associated with the LF node, and take the portion of the original sentence which led to it. This process is imperfect, and was intended as a quick way of recovering the answer. It tends to give phrases which span more words than necessary. For example, the LF node may describe an entity, but the corresponding syntactic node is a prepositional phrase, as the preposition is absorbed into the structure of the LF, resulting in an answer such as *by X* or *to X* rather than simply *X*. If the resulting phrase is longer than the maximum allowed width (50 bytes or 250 bytes), then words are removed from the ends of the phrase until it is short enough. By preference, words which appeared in the question are removed over ones which were not, and otherwise the process alternates removing words from the left and right hand ends of the phrase.

2.3 Answer selection

Answer selection is the heart of the matching algorithm. The rules used in the TREC-9 test are rather ad hoc; some of them are reasonably well principled, while others are hacks which seemed to work more often than any alternative. The principles we use to identify candidate answers nodes include the following:

Node properties
Node properties are used when answer nodes usually have clear LF properties, but where the relationship with the query terms can vary. ***Who***, ***HowMany*** and ***HowMuch*** questions are good examples, although we will see later that there is a risk involved in treating ***Who*** questions this way. The node properties are flags assigned by NLPWin usually using information stored in the lexicon. Node properties are used in three stages: firstly, we look for nodes which have one or more of a set of required properties; then we remove any which have certain properties which might indicate we have made the wrong choice as a result of over-generalisation; and finally, we look for preference properties whose omission indicates that the score assigned to the answer should be lowered. For example, in the case of ***Who*** questions, the only required property is ***PrprN*** (proper name), nodes are removed if they have properties such as ***Tme*** (time), ***Titl*** (title) and ***Cntry*** (country), and the score is lowered if node does not have one of the properties ***Anim*** (animate), ***Humn*** (human) or ***Nme*** (name).

Relation targets
Some answers can be found be looking for nodes which are the target of a given relation type, using proximity to determine whether the node is likely to be related to the question terms. Examples are ***Where*** and ***When*** questions, answers to which are often found as the target of ***Locn*** and ***Time*** relation. For ***When*** questions in particular, the answer time expression may appear on a different argument of a verb to the question term itself, or on a modifier of the question term.

Node-to-node relations
Node-to-node relations come closest to really using the structure of the LF. The idea here is to look for a node which lies at one end of a relation, the other end of which is a question term. The case where this is used most extensively is in questions of the form ***What is X***. Answers are typically found as standing in an *Equiv*, *Mod* or *Attrib* relation to *X* in the answer, as in the logical forms generated from phrases such as (the answer is highlighted):

Head Start is **a preschool program**
Berlin is the capital of Germany
Sirius, the brightest star visible from Earth

The first two of these illustrate *Equiv* (equivalent) relations, and show that the answer can be either the source or the target of the relation in this case. The third example is a *Mod* (modifier) relation. Some relations may signal the answer better than others; for example *Equiv* tends to indicate the answer more strongly than *Mod* or *Attrib* (attribute). The term which stands at the other end of the relation from the answer, i.e. the term from the question, may or may not be the head of the question. Thus if the question is ***What is the capital of Germany?***, the head of the question is *capital*, but we are as likely to find the answer related to the term *Germany*. Simple examples like this could be handled by specialised rules, for example manipulations of the question's LF, but this cannot always be done reliably. One case where we definitely do want the relation to be to a specific question word is questions about a specific role of a predicate. Thus, in ***Who won the SuperBowl in 1968?***, the answer should be in the subject role of the verb *win*.

Combinations

Some of the questions types use more than one of these techniques, and select the one which gave the best score. An example is *WhoRole* questions (which ask who performed a particular role of an action), which look for words with the same properties as *Who* questions, and also look for entities in a particular role of a verb, as for *WhRole* and *WhatRole* questions (node-to-node relation type of answers).

2.4 Proximity scoring

To assign a score to the nodes identified in answer selection, we use a simple measure based on how close the candidate answer is to significant terms from the question. The proximity measure marks each term in an answer sentence which matches a term from the question, and then sees how far this term is from the candidate answer, measured as the number of relations that have to be traversed in the logical form. The idea of proximity is to provide an approximation to matching the LFs, in that if an answer were closely related to the matched question terms, then it would have a small proximity, whereas if it had an indirect relation, the proximity would be lower. There is little linguistic basis for this approach, and the idea was really to obtain a baseline for performance based on a simple and easily implemented technique within the timescales of the TREC-9 exercise. The overall proximity is calculated by summing these distances for each of the question terms, taking its reciprocal, and weighting it by the logarithm of the total number of the matched question terms plus one. The latter factor is simply a way of taking into account what proportion of the question terms were matched. The logarithm is used just to weaken the factor; although this is ad hoc, it seems to give a better performance that using just the proportion of the query terms or no factor at all. An obvious enhancement to this process might be to weight question terms by importance, for example giving lower weight to question terms which are more deeply buried in the logical form.

3 Evaluation

3.1 The TREC-8 test set

The system was developed and tested using the questions and assessments from the TREC-8 evaluation. For an initial stage of evaluation, the retrieval stage was run in isolation, and the documents were examined to see if a correct answer appeared anywhere in them. This provides an upper bound on performance, by finding the best score which could be achieved if a perfect answer identification and extraction component were available. The evaluation also allowed tuning of the number of documents returned by Okapi: too few, and a correct answer might be missed; too many, and the processing time of the later stages would get out of hand. A document cutoff of 100 was selected for on this basis, which resulted in 92% of the questions retrieving a document which contained a correct answer. Larger cutoffs produced only a small further increase in this score. A variant of the experiment was run in which the term list for the retrieval was derived from the logical form, rather than by just taking the question and using a stemmed and stopped wordlist. The idea was to see if the segmentation and morphological analysis provided by NLPWin would help the retrieval stage. The scores were in general very slightly less than those above, showing that there is no clear advantage to using NLPWin as a pre-processor to the retrieval stage.

The performance for the overall system was calculated using mean reciprocal rank. Three scores were calculated: for the 50-byte and 250-byte limited runs, and for a run in the answer could be of any length, provided it lay within a single sentence. The results were as follows:

Run	50-byte	250-byte	Sentence
MRR	0.357	0.425	0.446

The first observation is that the best score, for the unlimited run, is significantly less than the maximum that could potentially be achieved with perfect answer selection (0.92, from the retrieval stage experiment). Secondly, the score does decrease with the window size, indicating that there is also scope for improvement in answer extraction. Compared to the official TREC-8, the 50 byte run would have come roughly 3rd out of 20 (or 21 including this run), and the 250 byte run about 10th out of 25.

3.2 TREC-9 test

The TREC-9 test consisted of 682 questions, including variant forms. The official evaluation results were:

Run	50-byte, strict	50-byte lenient	250-byte, strict	250-byte, lenient
MRR	0.196	0.203	0.264	0.274

Clearly, these are well below what we saw on the TREC-8 data. So what went wrong? In order to try to understand why, we look at the results for the separate question types in greater detail. In the table below, we list, for each question type:

- the number of questions of each type in the TREC-8 and TREC-9 test sets
- the MRR on that type

- the relative contribution of the class to the overall results
- the changes in MRR and relative contribution.

The relative contribution of a question type is the MRR for the type multiplied by the proportion of the questions which have that type. For example, if a type had a MRR of 0.5, and one quarter of all the questions had that type, the relative contribution would 0.5x0.25 = 0.125. The difference in MRR gives an indication in the abstract of the how well a question types was handled. If there is a large change, it would suggest that the rules for the type are too sensitive to the particular questions seen in the TREC-8 data. The change in relative contribution gives an indication of how much this matters, and therefore where efforts should be focussed to alter the system's performance. There may be more benefit in correcting a small decrease in MRR on a class with many questions as opposed to a large decrease on a class with only one or two. Some question types are handled identically, and we therefore list them both as the separate types and combined. The table is ordered by the change in the relative contribution, and the TREC-9 results are based on the 250-byte lenient judgements.

Question type	TREC-8			TREC-9			Change	
	#	MRR	Rel.Cont.	#	MRR	Rel.Cont	MRR	Rel.Cont
Unhandled	11	0.26	0.014	84	0.36	0.044	0.10	0.030
Unknown	2	1.0	0.010	38	0.37	0.020	-0.63	0.010
Bad	5	0.10	0.0025	6	0.50	0.0044	0.40	0.0019
WhPrep	3	0.11	0.0017	35	0.35	0.018	0.24	0.016
HowDo	1	0	0	5	0.20	0.0015	0.20	0.0015
WhoRole	20	0.29	0.029	62	0.36	0.032	0.073	0.0029
HowLong	1	0	0	1	0	0	0	0
HowManyTimes	1	0	0	0	0	0	0	0
WhatMeas	1	1.0	0.0050	8	0.29	0.0034	-0.71	-0.0016
When	18	0.38	0.034	47	0.45	0.031	0.070	-0.0032
Why	2	0.50	0.0050	2	0.5	0.0015	0	-0.0035
HowFar	1	1.0	0.0050	1	0	0	-1.0	-0.0050
WhatTime	6	0.29	0.0089	13	0.12	0.0022	-0.18	-0.0066
Where	21	0.44	0.046	71	0.37	0.038	-0.071	-0.0078
HowMuch	3	0.67	0.010	4	0.31	0.0018	-0.35	-0.0082
HowMany	15	0.28	0.022	26	0.29	0.011	0.0032	-0.010
HowProp	5	0.60	0.015	10	0.10	0.0015	-0.50	-0.014
What+	39	0.52	0.10	211	0.20	0.063	-0.22	-0.031
What	38	0.53	0.10	195	0.21	0.060	-0.32	-0.041
WhEquiv	1	0	0	16	0.15	0.0034	0.15	0.0034
WhRole+	28	0.38	0.053	92	0.16	0.021	-0.31	-0.038
WhRole	22	0.34	0.038	56	0.12	0.0095	-0.22	-0.028
WhatRole	6	0.50	0.015	36	0.23	0.012	-0.27	-0.0031
Who	28	0.55	0.078	52	0.30	0.023	-0.26	-0.055

It follows to look in more detail at what is going on in some of the more significant changes. Three classes in particular appear worth investigating on the basis of the change in relative contribution: *Who*, *WhRole+* and *What+*.

In the case of *Who* questions, the problem appears to be that some of the questions aim to identify an entity, while others aim to elicit a description of an individual. The two types are illustrated by

> Who is the richest person in the world? (entity)
> Who is Desmond Tutu? (description)

The TREC-8 test set included only entity questions, and the rules for answering *Who* questions did not allow for the description case. This could be corrected by adding a test to see if the question term already has the properties we look for in the entity case (*PrprN*, etc.), and if so using the same approach as *What* questions such as looking at *Equiv* and *Mod* relations.

The problem with *What* questions appears to be that many more of the questions have the form *What is the X of Y?* than the original set, for example,

> What was the name of the first Russian astronaut to do a spacewalk?
> What is the population of the Bahamas?

These are only handled well for a small number of predefined cases for the category condition *X*, such as *city*, *name*, and *kind*. To improve this class, we would need to have a set of additional rules which encode information about the category condition, for example that a population is usually expressed as a number.

A similar remark applies to the *WhRole* questions, many of which have the form *Which X does Y?*, such as

What sport do the Cleveland Cavaliers play?

Again, a few special cases are handled already, but the inclusion of some additional ones would help to select correct answers more reliably. One issue to be considered here is what conditions should have special rules and what should not. It is (perhaps) reasonable to have a list of sports for the above case, but what about

What soft drink would provide me with the biggest intake of caffeine?

The answer here appears to be some wider encoding of world knowledge. An interesting point emerges. If we are encoding world knowledge, should we try to encode all knowledge in the documents into some knowledge representation structure, and answer questions directly against it? This appears to be the thought process behind using MindNet (Richardson et al., 1998) in which dictionaries and encyclopedias are analysed and their logical forms merged into a single large structure, and it was also the approach used in the question-answering systems of the 1970s (Lehnert (1978), for example). The difficulty arises when the sources of the knowledge become more diverse and less coherent than those behind MindNet, or the Unix man pages used in ExtrAns (Aliod, 1998). There may be opinions, interpretations, inconsistencies, and simple errors in the document collection. An important challenge for future work may therefore be looking at how to build a system which merges definitive, pre-encoded knowledge, and ad-hoc documents of unknown reliability.

Appendix: Question types

These are the different types of questions which were used, with their frequencies in the TREC-8 and TREC-9 test sets.

HowDo (TREC-8: 1 TREC-9: 5) *How did Bob Marley die?*

HowFar (TREC-8: 1 TREC-9: 1) *How far away is the moon?*

HowLong (TREC-8: 1 TREC-9: 1) *How long do hermit crabs live?*

HowMany (TREC-8: 15 TREC-9: 27) *How many dogs pull a sled in the Iditarod?*

HowManyTimes (TREC-8: 1 TREC-9: 0)

How many times was pitcher, Warren Spahn, a 20-game winner in his 21 major league seasons?

HowMuch (TREC-8: 3 TREC-9: 4) *How much folic acid should an expectant mother get daily?*

HowProp (TREC-8: 5 TREC-9: 10) *How tall is the giraffe?*

WhEquiv (TREC-8: 1 TREC-9: 16) *What language is mostly spoken in Brazil?*

WhPrep (TREC-8: 3 TREC-9: 35) *What is Francis Scott Key best known for?*

WhRole (TREC-8: 22 TREC-9: 56) *What state has the most Indians?*

What (TREC-8: 38 TREC-9: 200)

What was the name of the first Russian astronaut to do a spacewalk?
What is Head Start?
Name a flying mammal.

WhatMeas (TREC-8: 1 TREC-9: 8)

What type of bridge is the Golden Gate Bridge?
What kind of animal was Winnie the Pooh?

WhatRole (TREC-8: 6 TREC-9: 36) *What does laser stand for?*

WhatTime (TREC-8: 6 TREC-9: 13) *What year did Montana become a state?*

When (TREC-8: 18 TREC-9: 48) *When did Vesuvius last erupt?*

Where (TREC-8: 21 TREC-9: 71) *Where is Belize located?*

Who (TREC-8: 28 TREC-9: 53) *Who is the leader of India?*

WhoRole (TREC-8: 20 TREC-9: 62) *Who invented the electric guitar?*

Why (TREC-8: 2 TREC-9: 2) *Why can't ostriches fly?*

Bad (TREC-8: 5 TREC-9: 6)

Questions which received no analysis from NLPWin, or a fragmentary one.

Unknown (TREC-8: 2 TREC-9: 39)

Other questions, not covered by any of the above classes.

Bibliography

Robertson, S. and Walker, S. (2001). *Microsoft Cambridge at TREC-9: Filtering track.* In these proceedings.

Bolc, Leonard (ed.) (1980). *Natural language question answering systems.* Macmillan.

Lehnert, Wendy G. (1978). *The process of question-answering: a computer simulation of cognition.* Erlbaum.

Molla Aliod, Diego, Jawad Berri and Michael Hess (2000). A real-world implementation of answer extraction. *Proc. of 9th International Conference and Workshop on Database and Expert Systems. Workshop "Natural Language and Information Systems" (NLIS'98).*

Stephen D. Richardson, William B. Dolan and Lucy Vanderwende (1998). MindNet: Acquiring and Structuring Semantic Information from Text. In *Proceedings of COLING.*

Microsoft Cambridge at TREC-9: Filtering track

S E Robertson* S Walker†

1 Summary

Apart from a short description of our Query Track contribution, this report is concerned with the Adaptive Filtering track only. There is a separate report in this volume [1] on the Microsoft Research Cambridge participation in QA track.

A number of runs were submitted for the Adaptive Filtering track, on all tasks (adaptive filtering, batch filtering and routing; three separate query sets; two evaluation measures). The filtering system is somewhat more advanced than the one used for TREC-8, and includes query modification and a more highly developed scheme for threshold adaptation. A number of diagnostic runs are also reported here.

2 Okapi at TRECs 1–8

A summary of the contributions to TRECs 1–7 by the Okapi team, first at City University London and then at Microsoft, is presented in [7]. Here we discuss only the routing and filtering task submissions.

Over the course of TRECs 1–6, we developed iterative methods of optimising the routing queries which were very successful, though computationally heavy. In successive TRECs our methods were enabled to explore more of the potentially huge space of possible queries. In TRECs 5 and 6 we used the same methods for batch filtering, again successfully.

However, we put these iterative methods aside for the adaptive filtering task in TREC-7. Here and in TREC-8 we concentrated on developing thresholding techniques, and did not in fact modify initial queries at all. This approach was relatively successful in TREC-7, but by TREC-8 many participants had better methods which additionally expanded or modified the queries adaptively, and we were somewhat left behind.

3 The system

At the Microsoft Research laboratory in Cambridge, we are developing an evaluation environment for a wide range of information retrieval experiments. This environment is called Keenbow. The Okapi BSS is seen as a component of Keenbow.

Many aspects of the system, including the weighting scheme and the query expansion methods used, reflect the various components of the probabilistic model of retrieval discussed at length in [8].

3.1 The Okapi Basic Search System (BSS)

The BSS, which has been used in all Okapi and Okapi/Keenbow TREC experiments, is a set-oriented ranked output system designed primarily for probabilistic-type retrieval of textual material using inverted indexes. There is a family of built-in weighting functions collectively known as BM25, as described in [6, Section 3] and subsequent TREC papers. In addition to weighting and ranking facilities it has the usual boolean and quasi-boolean (positional) operations and a number of non-standard set operations. Indexes are of a fairly conventional inverted type. There have been no major changes to the BSS during TREC-9.

3.2 Query expansion/modification

Given some known relevant documents, the query may be modified (primarily by adding new terms, but weights may be adjusted and an ineffective query term might also be dropped).

The initial step is to choose terms. Prior to TREC-8 the method used was that proposed in [9] by which terms are ranked in decreasing order of a term selection value or offer weight:

$$TSV = r.w^{(1)} \qquad (1)$$

(where $w^{(1)}$ is the Robertson/Sparck Jones weight [5], a component of BM25, and r is the number of (known) relevant documents in which the term occurs). The top t ranked terms are then chosen. For TREC-8 a new method was developed. This is based on a significance argument, and thus allows an absolute threshold on the offer weight, which may select different numbers of terms under different conditions. The formula is discussed in [7], and is as follows:

$$NTSV = r \log \frac{N}{n} - \log \binom{R}{r} - \log V \qquad (2)$$

---

*Microsoft Research Ltd, 1 Guildhall Street, Cambridge CB2 3NH, UK, and City University, London, UK. email `ser@microsoft.com`

†Microsoft Research Ltd, 1 Guildhall Street, Cambridge CB2 3NH, UK. email `sw@microsoft.com`

where r is as above; R is the total number of (known) relevant documents; n is the number of documents in the collection which contain the term; N is the size of the collection; V is the size of the vocabulary (number of distinct terms). We may use an absolute threshold criterion with this new offer weight:

$$NTSV > c \tag{3}$$

An argument was presented last year that zero would be a suitable value for c.[1]

The basic approach to query reformulation may now be described as follows:

1. extract all terms from all documents judged or assumed to be relevant
2. rank all terms, including original query terms, in order of offer weight
3. select those terms above a threshold or cut-off, defined as a threshold on the offer weight and/or a cut-off on the number of terms
4. weight the terms according to the usual relevance weighting formula (not the same as the offer weight)

Either or both the offer weight and the relevance weight may include some bias towards query terms; thus original query terms may remain in the query even if they occur in no or few relevant documents so far. However, the bias is not normally absolute: a query term which continues not to appear in relevant documents will eventually be dropped.

The above methods might be termed "model-based", and do not cover the iterative optimisation methods used in the routing task in earlier TRECs.

3.3 Filtering system

The filtering system used from TREC-7 on consists mainly of scripts built on top of the BSS.

The incoming "stream" of documents is divided fairly arbitrarily into batches (smaller batches initially to allow fast learning; larger later for efficiency reasons). For each topic a current state is maintained, including query formulation, threshold etc., what happened at the last batch, and some history, including docids for any documents judged relevant up to now. As a new batch of documents is processed, the current query formulation of each topic is searched against it; cumulative databases are created, and each topic goes through the adaptation process in preparation for the next batch. Adaptation includes query modification (term selection and weighting) and threshold adaptation; the various components are described below.

3.4 Hardware

All the TREC-9 processing was done at Microsoft Research, Cambridge. Most of the work was done on a 550MHz Xeon (512KB Cache) with 2Gb RAM and a Dell with two 400 MHz Pentium processors and 512 Mb. Both machines were running Solaris 7. The network was 100Mbps ethernet.

Table 1: Query track runs on Okapi

Query set	AveP	P@5	RPrec	Recall
acs1a	0.261	0.544	0.305	0.529
Sab1c	0.261	0.528	0.306	0.544
Titles	0.259	0.500	0.298	0.516
Sab1b	0.255	0.560	0.296	0.530
UoM2	0.254	0.564	0.305	0.573
pir1a	0.252	0.584	0.302	0.541
Sab1d	0.252	0.568	0.296	0.514
INQ1f	0.246	0.488	0.289	0.503
Sab1a	0.242	0.572	0.291	0.514
Sab2a	0.242	0.548	0.293	0.535
INQ1c	0.240	0.516	0.290	0.518
UoM1a	0.232	0.516	0.284	0.484
INQ2e	0.224	0.508	0.276	0.475
INQ1e	0.224	0.436	0.259	0.446
INQ2c	0.223	0.516	0.278	0.493
Sab3a	0.221	0.536	0.276	0.504
INQ1i	0.219	0.464	0.257	0.496
INQ1b	0.217	0.488	0.269	0.498
INQ1j	0.216	0.500	0.264	0.470
UoM1b	0.215	0.516	0.263	0.478
INQ1g	0.213	0.520	0.268	0.475
INQ1h	0.199	0.460	0.246	0.498
INQ1d	0.197	0.452	0.245	0.490
INQ2f	0.196	0.460	0.256	0.485
INQ2d	0.185	0.444	0.243	0.474
INQ1a	0.185	0.420	0.228	0.449
APL1a	0.182	0.432	0.233	0.433
INQ2g	0.180	0.428	0.242	0.423
INQ3e	0.175	0.432	0.214	0.440
APL2a	0.171	0.344	0.231	0.436
INQ2i	0.166	0.436	0.223	0.456
INQ2b	0.165	0.392	0.227	0.413
INQ2h	0.165	0.380	0.217	0.428
INQ2j	0.149	0.340	0.196	0.415
INQ3d	0.147	0.372	0.204	0.381
INQ3j	0.144	0.340	0.206	0.383
INQ3f	0.135	0.348	0.190	0.384
INQ2a	0.132	0.348	0.192	0.365
INQ3i	0.120	0.316	0.179	0.370
INQ3c	0.116	0.300	0.170	0.333
INQ3g	0.116	0.292	0.171	0.328
INQ3b	0.107	0.312	0.152	0.304
INQ3a	0.106	0.264	0.162	0.310
INQ3h	0.096	0.276	0.154	0.324

---

[1] The scale of this offer weight is $(-\infty, +\infty)$; a threshold of zero implies that we would expect about 1 noise term to be selected. We have discovered a bug in last year's programs, which means that last year's offer weights were offset by a certain amount; a correct zero threshold today is equivalent to a small negative threshold last year.

BEST COPY AVAILABLE

4 Query track

We did not take full part in the query track: that is, we did not generate queries. We did however run the queries that other participants had generated.

The system used to run these queries was an absolutely standard Okapi system, parsing the queries as provided in a standard manner and using BM25 weighting with $k_1 = 0.8$, $b = 0.4$, and $k_3 = 0$. No expansion was used. Some results for different query sets are shown in table 1, together with a corresponding run on topic titles only, sorted by average precision. Only two of the query sets outperformed topic titles on average precision, although several of them do better on other measures, particularly precision at 5 documents.

5 Filtering and routing

5.1 System design

For the last two years, the Keenbow/Okapi team has concentrated on the setting of thresholds for the adaptive filtering task. This year's effort is a much more rounded one, bringing together the thresholding methods and previously developed methods of query expansion and reweighting. At the same time, the introduction of the new target and measure into the adaptive filtering task has stimulated a significant expansion of the thresholding ideas, in a way which complements the previous approaches.

5.2 T9P thresholding: basic ideas

In the precision-oriented task, we have to attempt to retrieve the best 50 documents over the simulated life of the profile. The primary requirement is to set the threshold so as to retrieve close to that number of documents over the period (adjusting it as we go as necessary), while relying on the query to get us as close as possible to the best 50 documents.

Given a profile, some history of the stream of documents, and an expected rate of incoming new documents, we can relate the threshold to the number of documents in a model-free fashion, thus: we run the query against the accumulated collection so far, and rank the documents in the usual way; then the future number of documents whose score will reach a given threshold may be estimated from the number retrieved in the past at that threshold, adjusted pro-rata.

Such an estimate may not be very good, and will need adapting. So the principle is that after every batch of documents, we do a new retrospective search of the accumulated collection so far, and choose the threshold which is estimated to give us the right number of documents in the future, given what we have retrieved in the past. Since the evaluation measure penalizes under-retrieval more than over-retrieval, we aim a little higher than the nominal target of 50; in the current experiments, the margin is 25%, that is we aim for 62.5 documents. What happens if we hit the target before the end of the period is discussed below.

5.3 T9U thresholding: basic ideas

For the utility-oriented task, however, we go back to our work of TRECs 7 and 8. The basic requirement is to retrieve if the probability of relevance exceeds a certain figure; so we need a model to calibrate the score into a probability value. In TREC-7 we used quite a simple formula; in TREC-8 we tried something a little more complex, which gave us no performance improvement. This year we reverted to the TREC-7model.

The basic model for calibration is:

$$\log \frac{p_d}{1 - p_d} = \beta + \gamma \frac{s_d}{ast1} \tag{4}$$

where p_d is the probability of relevance of document d, s_d is its score, and $ast1$ is the average score of the top 1% of retrieved documents (actually, $ast1$ is in itself an example of model-free quantitative prediction). Initial values of β and γ were originally estimated from a logistic regression on old TREC data. For TREC-9, we simply re-used the TREC-7 initial values. Adaptation of β follows the method used at TRECs 7 and 8, summarized in the next section, and takes place after any new documents have been retrieved and/or the query has been reformulated.

Given a document score and an estimated $ast1$, equation 4 can be used to estimate the log-odds of relevance of any specific document. The calibrated score c_d is on a log-odds scale, but can be converted back to a probability p_d:

$$c_d = \beta + \gamma \frac{s_d}{ast1}; \quad p_d = \frac{\exp c_d}{1 + \exp c_d} \tag{5}$$

for some estimated β, γ and $ast1$.

As we obtain feedback, as well as re-estimating $ast1$, we adjust the calibration by correcting β (γ is left unchanged). We assume a set $\mathcal{F}$ of feedback documents whose relevance is known, of which r are relevant. A Bayesian prior is also assumed, represented by m mythical documents (in addition to those in $\mathcal{F}$), whose estimated probabilities of relevance are assumed to be correct at 0.5. We suppose an iterative sequence of estimates $\beta^{(n)}$ and corresponding values $c_d^{(n)}$ and $p_d^{(n)}$ for each document. Then the gradient descent formula is:

$$\beta^{(n+1)} = \beta^{(n)} + \frac{r - \sum_{d \in \mathcal{F}} p_d^{(n)} + m \frac{1 - \exp(\beta^{(n)} - \beta^{(0)})}{2(1 + \exp(\beta^{(n)} - \beta^{(0)}))}}{\sum_{d \in \mathcal{F}} p_d^{(n)} (1 - p_d^{(n)}) + m \frac{\exp(\beta^{(n)} - \beta^{(0)})}{(1 + \exp(\beta^{(n)} - \beta^{(0)}))^2}} \tag{6}$$

$\beta^{(0)}$ is the estimate of β taken from TREC-7.

In the last two TRECs, we ran this correction only once each time. Because the query may have changed

BEST COPY AVAILABLE

substantially since the last adjustment of β, we now (on each occasion we want to modify β) iterate the correction until the change is less than some small constant ϵ. Sometimes (after a substantial change in the query) the old β is badly out, and the gradient descent process becomes unstable. This can be resolved by setting a maximum correction to β in one iteration. In the experiments reported below, m is set at 3 (T9U runs) or 6 (T9P runs); ϵ is 0.01, and the maximum correction in one iteration is 1.0.

5.4 The cross-over: T9P task

A somewhat deeper analysis reveals an interesting cross-over between these two approaches of quantitative and qualitative prediction.

In the T9P task, we may reach the target before the end of the period. After this point the aim is to estimate the threshold score that will maximise the accumulated precision achieved at the end of the period. This requires both qualitative and quantitative prediction. The algorithm is essentially as follows:

1. perform a search with the current query on the accumulated collection so far, and rank the output
2. for the next document in this ranking, predict the number of documents achieving the same score in the future
3. predict the probability of relevance of these documents (from the score calibration)
4. estimate the overall precision that would be achieved if the threshold were set at this score
5. return to step 2
6. when the documents are exhausted, choose the score that gave the highest predicted overall precision as the threshold.

In the experiments reported below, this procedure is initiated when the total retrieved reaches 75% of the target. While the total remains less than the target, the rule is to aim for the target unless this procedure suggests trying for more documents. When the target is reached, then this procedure takes over. [2]

5.5 The cross-over: T9U task

In TRECs 7 and 8, we wanted to ensure that some documents were retrieved early on, even if their scores did not warrant it, in order to get some feedback to improve the query. The mechanism was a ladder of calibrated score values; a particular point on the ladder corresponded to the required utility, but we started lower down the ladder in order to get these initial documents. Both the ladder and the initial starting point were essentially arbitrary: we had no theory or mechanism to determine good values.

The quantitative approach now provides us with at least a way of thinking about the starting point. We would like to start in a position which would give us a small (non-zero) number of documents over the simulated period. The algorithm is essentially as follows:

1. calibrate the scores
2. determine the steps of the ladder
3. initially, or if we have not yet retrieved any documents,
    (a) estimate the threshold required to retrieve a certain target number of documents over the period
    (b) locate the ladder step closest to this threshold
4. if we have retrieved some relevant documents, then take a step up the ladder for every relevant document found so far (stopping at the top).

This procedure may be repeated at intervals. As soon as some documents have been retrieved, we stop being concerned about the target estimation, but remain on the ladder until we accumulate enough relevant documents to climb to the utility point. Because the ladder is defined in terms of the calibrated score, any intermediate stage that requires recalibration of the score (for example query reformulation) will be taken into account automatically.

The ladder currently in use is given in table 2.

Table 2: The Ladder

$P(R\|D)$	$\log O(R\|D)$	
0.33	-0.7	T9U
0.23	-1.2	
0.15	-1.7	
0.10	-2.2	
...	...	

The setting of an appropriate target number of documents is the subject of some of the experiments discussed below. It may also be noted that although there are still several arbitrary elements, this procedure should be a considerable improvement on our methods for TRECs 7 and 8, because the threshold will be set separately for each profile, in a way that relates to the particular query.

5.6 The accumulated collection

As in previous years, we assume that we accumulate the documents as they come into the system, so that we always have a cumulated collection of everything received up to now. Such a collection is needed for some of the forms of adaptation discussed; in the context of the TREC-9 filtering task, we actually need two such collections, respectively including and excluding the training set (Ohsumed 87).

---

[2] We have discovered a bug in this part of the program, which may cause the threshold to be set incorrectly if no relevant documents have been retrieved by the time we apply this procedure. The effect has not yet been investigated, but will be limited to a small number of topics.

BEST COPY AVAILABLE

5.7 Query reformulation

In the present filtering system, queries are reformulated as relevance information becomes available, as part of the adaptation process.

The method used is essentially that described in section 3.2. The new offer weight (equation 2) was used, with an absolute threshold. We *also* have a numerical term cut-off, which comes into effect when we have many relevant documents. We use this method right at the beginning, as our way of using the learning examples provided for the TREC-9 task. We repeat it at intervals determined by the retrieval of new relevant documents, frequently initially, and then only occasionally.

We set a limit on the number of relevant documents to be processed; if we have accumulated more than this number, we take only the most recent ones. This was implemented partly as an efficiency measure; however, it could be taken as a response to the possibility that either the user's interests, or the characteristics of the document stream, or both, may drift. In the present experiments, however, we have set this limit fairly high, so that it seldom comes into effect.

The principle tunable parameters of the query expansion method are (a) the maximum number of documents used (set to 100 here), (b) the term cut-off (maximum number of terms in the resulting query, 25 here), and (c) the absolute threshold on the offer weight (zero for these experiments). However, there are several other parameters or controls, e.g. the exact source and method of term extraction, and the form and degree of bias towards query terms.

5.8 Overview of the filtering procedure

At a particular iteration of the process, any query modification needs to take place before any threshold setting. It may also be necessary, after query reformulation but before threshold setting, to recalculate the scores of the previously-retrieved documents, for the adaptation of β.

As indicated in the system description, the incoming stream is batched somewhat arbitrarily, but with smaller batches initially on the grounds that the system needs to learn faster initially; later in the simulated period, profiles can be expected to have stabilized somewhat. In these experiments the test set (OHSUMED 88-91) is divided into 59 batches, initially 1000 documents per batch; the training set (OHSUMED 87) counts as batch 0.

For similar reasons, query modification is done after any batch in which a new checkpoint is reached for the particular topic. In these experiments, the checkpoints are defined in terms of the number of relevant documents retrieved so far, and are set at 1,2,4,8,16... relevant documents.

So the basic procedure is as follows: for each batch i of incoming documents

1. Run current profiles against batch i
2. Update both cumulative databases (batches 0–i and batches 1–i)
3. For each topic:
    (a) if checkpoint has been reached,
        - reformulate query
        - recalculate $ast1$ and scores of previously retrieved documents
        - re-estimate β using equation 6

    (b) set threshold (using methods described above)

6 Filtering and routing results

6.1 Topic sets

OHSU: 63 queries from the original OHSUMED queries, with relevance judgements from the requesters (no distinction was made between the two "relevant" categories)

MeSH: 4903 MeSH headings, treated as topics. The text of the topic is taken from the scope notes in MeSH; relevance judgements are the assignments of these headings to documents by the NLM indexers

MeSH-Sample: a sample of 500 of the MeSH topic set

6.2 Measures

T9U: linear utility, with relevant document credit set at 2 and non-relevant document debit set at 1, and a minimum utility of -100 for the OHSU topics and -400 for the MeSH topics.

MeanT9U: mean of T9U across topics, no normalisation

MeanSU: mean scaled utility across topics, where scaled utility is T9U divided by the maximum possible value for the topic, namely 2*(Total relevant)

T9P: precision, but with a minimum denominator of the target total number of documents retrieved, namely 50.

MeanT9P: mean of T9P values for each topic

MacR: macro average recall, that is the mean of recall values for each topic

MacP: macro average precision

and for routing runs, AveP (mean average precision) and P@50 (precision at 50 documents retrieved).

6.3 Submitted runs

See table 3. The rules for Batch and Routing allow the use of all the relevant documents in the training set for training, while for Adaptive Filtering 2 (OHSU) or 4 (MeSH) positive examples are provided. Those runs coded bfr or rfr did not make use of all the relevant documents, but only of all those retrieved in the top 100 documents in an initial search on the training set. (See next section for settings of some other parameters, which will explain the differences between some of these runs.)

Table 3: Submitted run results

Run	Type	Topics	Measure	MeanT9U	MeanT9P	AveP
ok9f1po	Adaptive	OHSU	T9P		0.294	
ok9f2po	Adaptive	OHSU	T9P		0.288	
ok9f2pm	Adaptive	MeSH	T9P		0.419	
ok9f1uo	Adaptive	OHSU	T9U	9.70		
ok9f3uo	Adaptive	OHSU	T9U	10.75		
ok9f1us	Adaptive	MeSH-Sample	T9U	46.53		
ok9f3us	Adaptive	MeSH-Sample	T9U	40.10		
ok9bf2po	Batch-adaptive	OHSU	T9P		0.305	
ok9bfr2po	Batch-adaptive	OHSU	T9P		0.305	
ok9bfr2ps	Batch-adaptive	MeSH-Sample	T9P		0.433	
ok9rf2po	Routing	OHSU				0.326
ok9rfr2po	Routing	OHSU				0.317
ok9rfr2ps	Routing	MeSH-Sample				0.245

6.4 Optimization runs

The system as described above contains a large number of settable parameters (tuning constants). Most of the parameters were set on the basis of guesswork, but some adjustments were made following some tests on some "pre-test" topics which had been provided. These pre-test topics are not actually supposed to be representative – indeed they consist of MeSH headings and OHSUMED queries which had been rejected from the main test for one reason or another, and exhibited some very different characteristics. Therefore this tuning process had to be a judicious mixture of experiment and guesswork. A very few of the parameters have been subjected to further testing, after the submission of the official runs, with the main test sets. These are reported here (table 4 and 5).

Note from Table 4 that there appears to be an optimum initial target for utility optimization, but that it depends on both the query set and the evaluation measure. It is higher for MeSH than for OHSU and higher for MeanT9U than for MeanSU. These results would be consistent with the hypothesis that the optimum depends on the number of relevant documents for the topic. The average number of relevant documents for MeSH topics is higher than for OHSU, and the MeanT9U measure is much more affected by topics with more relevant documents, while MeanSU weights all topics equally. But it seems that this difference cannot explain the full extent of the variation: the average number of relevant documents is approximately 50 (OHSU) and 250 (MeSH), but the difference in the optimum initial target is much greater. Another possibility is that the quality of the initial query is also important: a good initial query does not need much priming with relevant documents whereas a poor one does.

The possible effect of the total number of relevant documents raises the question of whether one might be able to make any useful kind of prediction of the optimum for a given topic. A plausible scenario for a real system would be to obtain an estimate from the user (which might or might not be good enough to help).

Table 4: Initial target for utility optimization

Target	MeanT9U	MeanSU	Notes
OHSU topics			
500	-2.37	-.410	
200	6.81	-.144	
150	8.86	-.082	
100	9.69	-.045	ok9f1uo
60	10.22	-.009	
30	10.75	.008	ok9f3uo
15	11.41	.029	
8	11.49	.032	
4	11.22	.033	
2	11.15	.033	
1	11.18	.034	One "zero return"
MeSH-Sample topics			
500	49.55	.076	
200	49.31	.099	
150	48.17	.102	
100	46.53	.102	ok9f1us
60	42.56	.101	
30	40.10	.098	ok9f3us
15	39.53	.097	

The "zero return" noted in Table 4 is a topic for which no documents were retrieved in the entire period. We regard this as a failure, but that run was the only one of the runs reported here that produced any zeros at all.

The data set is rather peculiar in terms of document length: about two-thirds of the documents have abstracts, while the other one-third do not; in the latter case, in effect the only text available is the title. There is therefore a huge discrepancy in document length between the two. *b* is the parameter in BM25 which controls the effect of doc-

Table 5: Document length

b	MacR	MacP	MeanT9P	Notes
OHSU topics				
0.8	.383	.288	.288	ok9f2po
0.4	.388	.294	.294	ok9f1po
MeSH-Sample topics				
0.8	.189	.430	.430	ok9f2ps
0.4	.181	.412	.412	

ument length. It was hypothesized that a high-precision task might benefit from concentrating on the documents with abstracts; reducing b would have that effect. Therefore in addition to the $b = 0.8$ value which is a good default, we tried a $b = 0.4$ run. This appeared to have some slight benefit with the OHSU topics, but the opposite effect in the MeSH topics (probably not significant).

Note also that the MacP and MeanT9P values are the same for each run. This reflects the success of the target setting and adaptation: either all topics retrieved over 50 documents, or the few which did not quite do so did not show up in the average. This is the case for all adaptive and batch-adaptive runs reported here, though not for non-adaptive batch runs.

6.5 Some comparisons

The new T9P measure provides an interesting opportunity to compare the results of filtering runs with traditional ranked-retrieval runs. The evaluation program trec_eval for ranked retrieval calculates P@n values – precision at n documents retrieved – for various values of n. T9P is a sort of P@50[3]. However, the comparison needs to be qualified, as discussed in the overview paper [12].

This analysis concentrates on the effect of using different amounts of relevance information at different stages. For the routing results, we have pure P@50 on the test set (we include AveP in the table also); no threshold is involved. For batch filtering (non-adaptive), we use exactly the same queries as for routing, but set a once-only threshold intended to retrieve 50 documents. For batch-adaptive or adaptive filtering, we may either adapt only the threshold, or we may adapt both the threshold and the query. All of these may be done starting with all relevant documents in the training set, or with only those which would have been found in an initial search, or with the 2 or 4 positive examples provided for adaptive filtering, or with none. All these conditions are represented in table 6. In the last two columns, MacP figures are in all cases the same as the corresponding MeanT9P figures given. The adaptive filtering rules allowed 4 training relevants for MeSH topics; we have included a '2 relevants' row for comparison with the OHSU topics. The two were chosen as the first two of the four provided.

Within each topic set, P@50 reflects AveP quite closely.

Comparing MacP and MeanT9P figures for the 'No adaptation' column, we see that MacP is consistently higher than MeanT9P. This reflects two factors: first, there is a lot of variation in the number of documents retrieved, so that many topics failed to retrieve 50 documents. Second, the initial threshold is often too high. Some further experiments suggest that query adaptation to the relevant documents in the training set tends to interfere with the initial threshold setting to cause this effect.

Comparing the last two columns for the 'All relevant' training, we notice a small decline in performance. The 'all relevant' set may be seen as an unbiased sample of relevant documents; the extra relevants used to modify the query during adaptation are to some extent biased towards the query. However, modifying the query becomes progressively more useful as we start from less relevance information, and adapting the threshold appears always to be beneficial.

There seem to be some differences between the two topic sets as to how useful each level of relevance information is. This may perhaps reflect two things: differences in the quality of the initial queries and differences in the total number of relevant documents per topic.

Comparing P@50 with final MeanT9P in the '2/4 relevants' case, we see that full adaptation just about compensates for the inherent difficulty of MeanT9P.

A different kind of analysis may be made by considering the utility measure. We regard T9P as a high-precision task; however, in order to score above zero on T9U we have to obtain a precision of at least 33%. The difficulty of this task is reinforced by looking at the precision values obtained for T9P runs. For example, in the case of OHSU, these seldom reach 33%. Linear utility (with the sort of parameter values used for the last 3 years) is indeed a hard, high-precision task, and it is not so surprising that we had such difficulty in doing even reasonably well at it.

6.6 Computational load

Running the 60 batches and 4900 MeSH topics is a heavy computational task. Although each batch is quite small, it involves both the 4900 basic searches and all the additional work (including possibly more than one search on the accumulated collection) required for adaptation of a topic, again 4900 times. The scripts used for this task take approximately one week to run on the 550 MHz, 2Gb machine. They could no doubt be made considerably more efficient; nevertheless, the adaptive filtering task must be regarded as computationally heavy – considerably more so than, say, the 100Gb VLC or Web track.

---

[3] trec_eval does not by default include $n = 50$; however, a simple modification of a header file allows it

BEST COPY AVAILABLE

Table 6: Relevance information and adaptation

	No threshold			No adaptation		Threshold adaptation	Threshold and query adaptation	
Training	AveP	P@50		MacP	MeanT9P	MeanT9P	MeanT9P	
OHSU topics								
All training set relevants	.326	.336	ok9rf2po	.357	.274	.309	.305	ok9bf2po
Relevants in top 100	.317	.324	ok9rfr2po	.342	.266	.296	.305	ok9bfr2po
2 training relevants	.277	.294		.281	.251	.268	.288	ok9f2po
No relevants	.228	.260		.240	.221	.236	.280	
MeSH-Sample topics								
All training set relevants	.283	.490		.506	.412	.472	.461	
Relevants in top 100	.253	.455	ok9rfr2ps	.458	.366	.429	.433	ok9bfr2ps
4 training relevants	.245	.437		.428	.375	.413	.430	
2 training relevants	.201	.397		.385	.344	.373	.415	
No relevants	.135	.301		.290	.262	.285	.390	

7 Conclusions

The adaptive filtering task continues to be an interesting and fruitful one to investigate. The new T9P optimization measure has been very successful, both in encouraging the development of the thresholding methods (which are now much stronger for both measures), and in allowing some comparison of traditional ranked-retrieval performance with threshold-based filtering.

References

[1] Elworthy, D. Question answering using a large NLP system. In these proceedings. 2001.

[2] Mitev, N.N., Venner, G.M. and Walker, S. *Designing an online public access catalogue: Okapi, a catalogue on a local area network.* British Library, 1985. (Library and Information Research Report 39.)

[3] Walker, S. and Jones, R.M. *Improving subject retrieval in online catalogues: 1. Stemming, automatic spelling correction and cross-reference tables.* British Library, 1987. (British Library Research Paper 24.)

[4] Walker, S. and De Vere, R. *Improving subject retrieval in online catalogues: 2. Relevance feedback and query expansion.* British Library, 1990. (British Library Research Paper 72.) ISBN 0-7123-3219-7

[5] Robertson, S.E. and Sparck Jones K. Relevance weighting of search terms. *Journal of the American Society for Information Science* 27, 1976, p129–146.

[6] Robertson, S.E. *et al.* Okapi at TREC–3. In: [10], p109–126.

[7] Robertson, S.E. and Walker, S. Okapi/Keenbow at TREC-8. In: [11], p151–162.

[8] Sparck Jones, K., Walker, S. and Robertson, S.E. A probabilistic model of information retrieval: development and comparative experiments. *Information Processing and Management*, 36, 2000, p779–808 (Part 1) and 809–840 (Part 2).

[9] Robertson, S.E. On term selection for query expansion. *Journal of Documentation* 46, 1990, p359–364.

[10] *Overview of the Third Text REtrieval Conference (TREC–3).* Edited by D.K. Harman. Gaithersburg, MD: NIST, 1995 (NIST Special Publication 500-225).

[11] *The Eighth Text REtrieval Conference (TREC–8).* Edited by E.M. Voorhees and D.K. Harman. Gaithersburg, MD: NIST, 2000 (NIST Special Publication 500-246).

[12] Robertson, S.E. and Hull, D.A., The TREC–9 Filtering Track final report. In these proceedings. 2001.

Another Sys Called Qanda[1]

Eric Breck, John Burger, Lisa Ferro,
Warren Greiff, Marc Light,[2]
Inderjeet Mani, Jason Rennie

The MITRE Corporation
202 Burlington Road
Bedford, MA 01730

Introduction

This year our primary goal was to improve on the performance of our TREC-8 system. In addition to improving the system directly, we worked on a number of tools to aid our development. We continued our work on a tool for automatic scoring of system responses, a "judge" program. We designed a tool for doing regression testing of question answering systems. We developed a measure of candidate confusability which measures the effectiveness of a set of features for reducing the choices that a ranking system has to make: a coarse form of perplexity. Finally, we performed preliminary work on a method for generating supervised training data.

We began with our system from the TREC-8 competition (Breck et al., 1999). Like many of the TREC-8 systems, it had the system design illustrated in Figure 1. The input question is processed by a question analyzer, which assigns it one of several dozen answer types. The question is also fed to an information retrieval engine, which returns a set of documents. Next, a set of taggers finds entities of the type assigned by the question analyzer and other related types. These entities are then ranked as to how likely they are to be the answer. Finally, answer strings are generated for the top candidates.

For this year, we kept this basic design but improved many of the modules involved. For example, we extended our answer type inventory, improved the question analyzer, and added a temporal dereferencing module to the taggers (Mani & Wilson, 2000). However, much of our effort went into the candidate ranking system. We did this for two reasons. First, an error analysis last year showed that problems in candidate ranking caused a substantial portion of Qanda's error (21%). Second, it seemed to be an appropriate place for a principled method of

---

[1] In keeping with our marine theme, we considered renaming our system Flounder due to its poor performance this year. A trivial bug in the answer candidate ranking system caused candidates to be ranked essentially at random. Perhaps this run should be considered a "chance" baseline.

[2] Principal contact: <light@mitre.org>

The performance of A1W model is almost the same as CO model. The average precision of A1W model is 0.0787 that is 32.9% of monolingual information retrieval. When we augmented restriction terms to an original query term, we also added noises. On the other hand, some good terms were not added. Recall that we only augmented co-occurrence terms that have only one translation. Many good terms are ambiguous so that they cannot be added. For example, since 'programmer' has four translation equivalents, it could not be selected as a restriction for 'computer'.

4. Conclusion

For the QA Track, we proposed three models this year. These models can help us to see the usefulness of each proposed factor. Base Model uses the information of named entity and its equivalence, as well as the information of inflection forms of general nouns and verbs. Synonyms of nouns and verbs are proved to be of little use. Simple co-reference resolution causes a drawback because of the wrongly merged passages.

At the Cross-Language Track, we proposed two models to translate queries: CO and A1W model. Two runs were submitted to cross-language track, and the performances are not so good as our expectation. The major reason is that our IR system has bugs. After correcting the bugs, we redid the experiment. The average precisions of CO and A1W model are 0.0784 and 0.0787 respectively. The improvement of A1W model was limited. How to select augmented restriction terms is a problem. We will make more experiments to see what strategy is more appropriate.

Reference

Bian, G.W. and Chen, H.H. (1998) "Integrating Query Translation and Document Translation in a Cross-Language Information Retrieval System." *Machine Translation and Information Soup*, Lecture Notes in Computer Science, No. 1529, Spring-Verlag, 250-265.

Chen, H.H., Bian, G.W. and Lin, W.C. (1999) "Resolving Translation Ambiguity and Target Polysemy in Cross-Language Information Retrieval." *Proceedings of 37th Annual Meeting of the Association for Computational Linguistics*, 215-222.

Church, K. *et al.* (1989) "Parsing, Word Associations and Typical Predicate-Argument Relations." *Proceedings of International Workshop on Parsing Technologies*, 389-398.

Harman, D.K. (1997) *TREC-6 Proceedings*, Gaithersburg, Maryland.

Huang, C.R., *et al.* (1995) "Introduction to Academia Sinica Balanced Corpus." *Proceedings of ROCLING VIII*, Taiwan, 81-99.

Lin, C.J. and Chen, H.H. (1999) "Description of Preliminary Results to TREC-8 QA Task." *TREC-8 Proceedings*, Gaithersburg, Maryland, http://trec.nist.gov/pubs/trec8/papers/NTU_TREC8_QA.pdf

Miller, G. (1990) "Five Papers on WordNet." *Special Issue of International Journal of Lexicography* 3.

MUC (1998) *Proceedings of 7th Message Understanding Conference*, http://www.muc.saic.com/proceedings/proceedings_index.html.

Salton, G. and Buckley, C. (1988) "Term Weighting Approaches in Automatic Text Retrieval." *Information Processing and Management*, Vol. 5, No. 24, 513-523.

NTT DATA TREC-9 Question Answering Track Report

Toru Takaki
Research and Development Headquarters
NTT Data Corporation
Kayabacho Tower Bldg., 1-21-2, Shinkawa,
Chuo-ku, Tokyo 104-0033 Japan
E-mail: takaki@rd.nttdata.co.jp

Abstract

This paper describes the processing details and TREC-9 question answering results for our QA system. We use a general information retrieval strategy and a simple information extraction method with our QA system. Two types of indices, one for documents and one for passages, were used for our experiment. We submitted four results, two for each category of short and long answers. A score of 0.231 for the short category and 0.391 for the long category was obtained.

1 Introduction

Question-Answering (QA) processing has been attracting a great deal of attention recently. This type of retrieval processing requires the use of techniques to retrieve pertinent information from within a document that differ from those used for document retrieval. A QA system that can retrieve concise and suitable information that satisfies the needs of users will contribute to the improvement of recall precision and enhance a user's productivity when they are searching for information using the vast and ever expanding resources of the worldwide web. The currently used document retrieval method, which outputs a document list, forces users to search individual documents to find the information they desire. The Text REtrieval Conference (TREC) is designed the QA Track as one of tracks from TREC-8 in 1999. We regard QA as an application that unites natural language processing, information extraction, and information retrieval processing, which is a traditional and refined technology that is based mainly on the frequency of term occurrence. This traditional information retrieval technology and natural language processing, based on semantics, are indispensable to the QA processing.

We participated in the TREC-9 Question-Answering track held this year. This was the second time we have participated in a QA track (TREC-8 was the first). For this track, we combined the traditional information retrieval technique and the information extraction technique to construct a QA system. Our TREC-9 QA system was based on our TREC-8 QA system to which we had made some improvements. Our official runs at the TREC-9 QA were executed by changing some parameters and the units of the index, document, or passage to be retrieved in the initial retrieval processing that are applied in traditional information retrieval. In this report, our QA system is described along with an analysis of the initial search-processing step, and the results of our official runs are shown.

2 Processing flow

This section describes the processing flow of our QA system. The processing was done according to the four steps below (see Figure 1):

(1) Question analysis
The answer type is specified by an analysis of question. Then, query terms for the initial search are extracted from the question sentence,

BEST COPY AVAILABLE

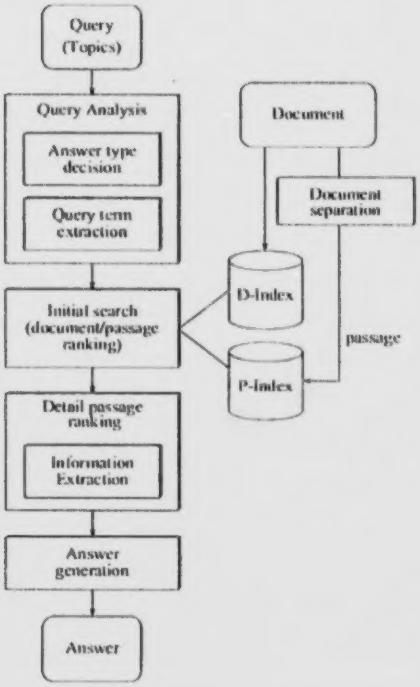


Figure 1: Processing flow

(2) Initial search
An initial search is done to limit the number of documents searched in the next step of the processing. The traditional ranking method, based on term frequency, is applied to the initial search,

(3) Detail passage ranking
Selection of important passage-spans and their rankings are done. The passage-span ranking uses the information extraction results of the specified answer type, and

(4) Answer generation
To provide an answer within the restricted length, 50 bytes or 250 bytes, the answer is extracted from the top-ranked passage.

Processing details are described below.

2.1 Answer type specification

The answer type specification is a processing step in which a determination is made as to what type of answer is required for a given question (topic). This specified answer type is used to extract the answer from the document in the next step. In the answer type specific processing, we created question templates that defined the answer type for question phrases, such as a ***wh-determiner***, a ***wh-pronoun***, etc. For instance, Topic No.206 "***How far away is the moon?***" suits the template "***How far ....***" , so the answer type is determined to be LENGTH. We defined 28 answer types, as shown in Table 1. These answer types have a hierarchical structure. Therefore, one question is will not always have only one answer type; sometimes two or more answer types can be given. In the case of Topic No.271 "***How tall is a giraffe?***", the prime answer type candidate is LENGTH and the second candidate is NUMBER, that is in a high-ranking hierarchy. The determination of whether or not the answer type of a high-ranking hierarchy is to be an answer type candidate is made based on a consideration of the question template. Moreover, questions that have no template match are given UNDEFINED as their answer type. Our template does not have provisions for a why-question.

2.2 Query term extraction

The query terms are extracted from the question, and used to search the candidate documents in document database. The purpose of this search is to minimize the number of the candidate documents. The high-cost processing executed later, such as passage ranking and information extraction, is done for only documents where the probability is high that they include the correct answer. The search for ranking is based on the frequency of the query term, like "ad-hoc" retrieval, and the top-ranked document is the candidate having the correct answer. However, the query term expansion processing usually done in an "ad-hoc" retrieval is not performed in our system.

Top level	Middle level	Bottom level
PROPER	PERSON	CHAIRMAN
		LEADER
		MINISTER
		PRESIDENT
		SECRETARY
		SPECIALIST
	LOCATION	CITY
		COUNTRY
		STATE
	COMPANY	
	LAKE	
	RIVER	
	MOUNTAIN	
	LANGUAGE	
NUMBER	SIZE	
	LENGTH	
	MONEY	
	PERCENT	
	PERIOD	
TIME	DATE	
	YEAR	
UNDEFINED-PROPER		
UNDEFINED		

Table 1: Answer type

Deletion of stop words

Unnecessary terms were deleted from a question in accord with a 550-stopword list.

Extraction of multiword phrase

A multiword phrase was extracted by using a part-of-speech tagger and then used as the query term. Each single-word term, which was parts of the multiword phrase, was also made into a query term.

Extraction of preposition phrase

In the QA retrieval, some questions required limited information. Topic No.32 used in TREC-8 QA "*What is the largest city in Germany?* " required the "*largest city in Germany*". If "*in Germany*" is not extracted as a query term, other "*largest city*", such as "*in the world*" or "*in Japan*" etc., cannot be distinguished without "*in Germany*". Therefore, the preposition phrase is important in the QA retrieval. Thus, the preposition phrase was made a query term.

Extraction of quotation phrase

Since a quotation phrase is a limiting expression that is close to the content of a question, like a preposition phrase, we adopted it as a query term.

Query term's weighting

The degree of importance was given to a basic word, a multiword phrase, a quotation phrase and a preposition phrase. The multiword phrase is divided into single-words, and both the single-words and the multiword used as query terms.

Unit of index of retrieval document

The document set used for TREC-9 consists of the following data sets from the TIPSTER and TREC document CDs:

AP Newswire (Disks 1-3),
Wall Street Journal (Disks 1-2),
San Jose Mercury News (Disk 3),
Financial Times (Disk 4),
Los Angeles Times (Disk 5), and
Foreign Broadcast Information Service (Disk 5).

In our experiment, the following two units were used as an index for the initial search.

(1) Original document (data was the part enclosed with <DOC >and </DOC >) 978,952 documents of TREC-9 evaluation used as index units.

(2) Paragraph divided the original document The unit of division was different depending on the kind of the document. More than 978,952 documents were divided into 11,343,632 parts.

The QA track required the extraction of the pertinent answer, not the unit of document. So the division of document into a paragraph by paragraph ranking made it possible to extract a more suitable answer. Each paragraph is given an identifier, such as AP90424-0079-000046, that are a combination of the paragraph identifier (000046) and the document identifier (AP900424-0079).

Ranking of initial search

The query terms and their weights are input into the initial search. Both the document and the paragraph are ranked according to the input. In our QA system, we did the relevance ranking of a document or a paragraph using the BM25 function of Okapi. This function is as follows:

$$\sum_{T \in Q} w^{(1)} \frac{(k_1 + 1)tf}{K + tf} \frac{(k_3 + 1)qtf}{k_3 + qtf} \qquad (1)$$

where
Q is a query, containing terms T,
$w^{(1)}$ is the Robertson/Sparch Jones weight of T in Q,

$$w^{(1)} = \frac{(r + 0.5)/(R - r + 0.5)}{(n - r + 0.5)/(N - n - R + r + 0.5)} \qquad (2)$$

N is the number of documents/paragraphs in the collection,
n is the number of documents/paragraphs containing the term,
R is the number of documents/paragraphs known to be relevant to a specific question,
r is the number of relevant documents/paragraphs containing the term,
K is $k_1((1 - b) + b \times dl/avdl)$,
tf is the frequency of occurrence of the term within a specific question,
qtf is the frequency of the term within the question from Q was derived, and
dl and $avdl$ are the document length and average document length.

The documents or paragraphs that ranked in this ranking process are considered to include the correct answer. Therefore, for the subsequent processing, we used only the top-ranked documents from the document ranking and the documents that included the top-ranked paragraph.
Both the top m_d documents from the document ranking and the top m_p documents of the paragraph ranking were assumed to be the following processing object. Even if there is an overlapping of the top m_d and m_p documents, m_d and m_p are not changed. The number of following processing documents to be processed subsequently is assumed to be M.

2.3 Passage ranking and information extraction

The candidate passages that may include the answer are specified and extracted from the M top-ranked documents obtained in the previous step. These passages are part of the top-ranked documents, and the part has much the query words/terms and the words/terms matched the answer types. By using this concept there is a high probability of finding the correct answer. This passage extraction method is based on traditional information retrieval techniques, such as relevance ranking. A passage was extracted using the following procedure:

(1) Scoring by query term
A score was given to each word of the top-ranked document. This score was based on the inversed document frequency (IDF) measure of query term q_i. When each word of document D was assumed, the word P_i (i = 1,2,3,...) from the top of the document in ascending order, a score, $IDF(q_i)$, given to word P_j where query term Q had appeared. Moreover, score IDF' is given to the word of P_k at the circumference term position of P_j, and this score was based on the distance from P_j. The longer the distance from P_j, the smaller the score given to the circumference word. In some cases, where there were two or more query terms in the question sentence, a term score was given to each query term, and the sum of the score given by each query term was assumed to be the score for P_j. The consecutive passages where the score was more than the threshold were determined to be candidate passages.

(2) Scoring by answer type
Same as the scoring by query term. The bonus score for each word in the extracted passage was given by the words of the answer type. The answer

type was given one or more candidate types in the order of importance.

The word of these answer types was extracted by the information extraction technique, and the bonus score was given to the word. When the maximum value of the bonus score of the word of the prime candidate's answer type is S, S/k was given to the word of the k-th candidate's answer type. In our information extraction, we prepared proper name dictionaries, such as country, city, world region, U.S. state, currency, a personal name, and the dictionaries of literal form, such as date and time.

2.4 Answer generation

An answer generator outputs the answer string within the restricted length number, from 50 up to 250 bytes. The region, which included the word having the highest score in the passage, was outputted as the answer. The system did not output the string same as the term within the question.

3 Analysis of initial search

In this section, we analyze the initial search, which is one of QA retrieval steps used by the topics of the TREC-9 QA track as evaluation data. The initial search is based on a traditional information retrieval technique. Here, we analyze the change of the initial search accuracy by using the index for document or paragraph units.

Initial search

The QA initial search is a relevance ranking of the document/paragraph used with the query terms that are extracted by using the question sentence. In the QA retrieval, some natural language processing and information extraction processing are necessary, and the cost of this processing is needed. Placing restrictions on the amount of data to be searched is useful from the viewpoint of the processing speed, especially when retrieving a huge amount of data. Moreover, the use of traditional information retrieval technology is also beneficial. However, there is a method of whereby the information extraction result can be put in the index beforehand. In this method, if the information extraction module is imperfect, the information extraction processing for all data to be indexed must be done after the module is corrected. Therefore, we adopted this method of initial search.

Initial search accuracy by difference of index

We analyzed the initial search ranking that show how many document had the correct answer in the document top-ranked by the TREC-9 QA question and dataset. In this analysis, we used the TREC-9 QA judgment file provided by NIST, the top 1000 ranked document results ranked by the AT&T version of SMART provided by NIST, and our initial search results that were used for our submitted systems. We investigated the highest ranked document that included the correct answer, outputted by each system's initial search for each TREC-9 QA question. High precision is required in a QA retrieval, especially so in the rules of the TREC QA (it is not required that a system output all the correct answer phrases in a document). In addition, this analysis becomes the indicator of the threshold decision for how many top-ranked documents should be used to obtain the highest accuracy. The initial search retrieval results of our system and the SMART system were examined and the highest ranking, which contained the correct answer for each question, were examined. Table 2 shows the number of the question at the highest rank that included the correct answer for 682 TREC-9 QA topics. Here, the percentage shows the accumulation ratio of a ranking.

We prepared an index of both the unit of the document and for each paragraph so as to perform a comparison. NTTD-D means by document index and NTTD-P means paragraph index. In NTTD-D, the document of 48.2%, 67.7%, 75.7%, and 80.5% contained the correct answer of a question at the rankings of 1,3,5, and 10. Even for rank 5, the rising degree of the accumulation ratio was high but the rising growth was lower at the lower ranking. The tendency of SMART was also similar. Moreover, the retrieval accuracy of document index (NTTD-D) was better than that of the para-

Highest rank	SMART		NTTD-D		NTTD-P	
	#Q		#Q		#Q	
1	287	(42.1%)	329	(48.2%)	279	(40.9%)
2	63	(51.3%)	83	(60.4%)	90	(54.1%)
3	39	(57.0%)	50	(67.7%)	45	(60.7%)
4	33	(61.9%)	39	(73.5%)	16	(63.0%)
5	33	(66.7%)	15	(75.7%)	29	(67.3%)
6	19	(69.5%)	6	(76.5%)	13	(69.2%)
7	10	(71.0%)	11	(78.2%)	10	(70.7%)
8	7	(72.0%)	6	(79.0%)	7	(71.7%)
9	5	(72.7%)	6	(79.9%)	15	(73.9%)
10	3	(73.2%)	4	(80.5%)	11	(75.5%)
11-20	39	(78.9%)	31	(85.0%)	42	(81.7%)
21-30	21	(82.0%)	18	(87.7%)	21	(84.8%)
31-40	18	(84.6%)	11	(89.3%)	11	(86.4%)
41-50	10	(86.1%)	12	(91.1%)	9	(87.7%)
51-60	7	(87.1%)	6	(91.9%)	3	(88.1%)
61-70	4	(87.7%)	6	(92.8%)	6	(89.0%)
71-80	4	(88.3%)	3	(93.3%)	1	(89.1%)
81-90	3	(88.7%)	3	(93.7%)	2	(89.4%)
91-100	2	(89.0%)	1	(93.8%)	4	(90.0%)
other	75	(100.0%)	42	(100.0%)	68	(100.0%)

Table 2: Highest rank of initial search

graph index (NTTD-P) in the comparison of the initial search. In our system, the parameter setting of the BM25 function for the document index did not change for paragraph index in the initial search. Therefore, it was thought that this was the reason for the decrease in accuracy for the paragraphs. However, we did not do a detailed analysis. Moreover, it would have been necessary to analyze whether to or not the paragraph division was done correctly. We set the threshold, that is the number of the document to be used for processing after initial search, to 5 or less in our TREC-9 QA system, and the used document is very limited.

4 TREC-9 evaluation result

We submitted four results in TREC-9 QA track; there are two results each for the 50-byte answer and 250-byte answer categories. **NTTD9QAa1S** and **NTTD9QAa2S** are run names for the 50-byte category, and **NTTD9QAa1L** and **NTTD9QAb1L** are for the 250-byte categories. The difference for each run are the index used and the number of the top-ranked document used for the detail passage ranking in the initial search. As mentioned above, the sum document of m_d, from the document index, and m_p, from the paragraph index, were used as candidate documents to do the detail passage ranking. The parameter was $m_d = 3$ and $m_p = 2$ in **NTTD9QAa1S** and **NTTD9QAa1L**, $m_d = 5$ and $m_p = 3$ in **NTTD9QAa2S**, and $m_d = 3$ and $m_p = 0$ in **NTTD9QAb1L** (using only the document index and not the paragraph index). The other processing was the same for each run. Table 3 summarizes the evaluation results provided by NIST for our system. The results show that our mean reciprocal rank (MRR), except **NTTD9QAa1S**, was better than the average of all participants. We calculated the difference of MRR with **NTTD9QAa2S** and the average for all participants in the 50-byte category, and analyzed our system with having large MRR difference, named

Run tag name	Mean reciprocal rank (MRR) [Average MRR]	Num. of answers found at rank X						#Q	#Q
		1st	2nd	3rd	4th	5th	Not found	Best	≥ Med
NTTD9QAa1S	0.216 [0.218]	103	49	34	23	12	461	158	597
NTTD9QAa2S	0.231 [0.218]	108	61	24	26	24	439	161	599
NTTD9QAa1L	0.391 [0.350]	191	95	51	35	11	299	208	541
NTTD9QAb1L	0.381 [0.350]	195	79	40	26	29	313	212	534

Table 3: Our submitted TREC-9 QA runs

	Topic No.	MRR of **NTTD9QAa2S**	MRR of Average	Difference	Question
1	817	1.000	0.030	0.970	*Boxing Day is celebrated on what date?*
2	633	1.000	0.061	0.939	*How long do hermit crabs live?*
3	490	1.000	0.061	0.939	*Where did guinea pigs originate?*
4	541	1.000	0.061	0.939	*What was the purpose of the Manhattan project?*
5	779	1.000	0.080	0.920	*Name the university of which Woodrow Wilson was president.*
6	731	1.000	0.091	0.909	*What amount of folic acid should an expectant mother take daily?*
7	383	1.000	0.098	0.902	*What is the largest variety of cactus?*
8	661	1.000	0.119	0.881	*How much does one ton of cement cost?*
9	398	1.000	0.121	0.879	*When is Boxing Day?*
10	815	1.000	0.121	0.879	*What is the date of Boxing Day?*

Table 4: Best results and questions

as best and worst, shown in Tables 4 and 5.

First, we analyzed the answer type decision procedure. When our best and worst were compared, there were a lot of "*where*" questions in the worst. In the "*where*" question (71 questions), **NTTD9QAa2S** was 0.235 against the average MRR of 0.314. The reason for this low performance, determined by a detailed analysis of the "*Where*" question results was that either the answer types CITY, COUNTRY, or STATE were judged as a LOCATION. LOCATION is a more abstract answer type. Therefore, another feature extraction that can judge in detail and another answer type should be added to our system. However, as for 59 questions judged to be answer type NUMBER, **NTTD9QAa2S** result was excellent; the MRR was 0.260 vs. 0.201 for the average MRR. Next, the initial search result was analyzed. It was apparent that all top-ranked documents of **NTTD9QAa2S**'s initial search included the correct answer in the best case. For the worst case, we examined the 10 worst questions and found only one question for which the initial search failed to give a document including the correct answer a high ranking to document. This shows that our system failed in either the passage ranking or answer generation steps. In the case of Topic No.614 "*Who wrote the book, "Huckleberry Finn"?*", the correct answers are "*Samuel Langhorne Clemens*" and "*Mark Twain*". The correct answer was included in a document of the second rank in an initial search of NTTD-D and of the first rank in NTTD-P. However, our system has a problem in that it is not able to extract the correct answer phrase in the 50-byte answer category when the answer appeared at a position away a little. This reason is that our system emphatically determined the important part of a passage using the appear-

	Topic No.	MRR of NTTD9QAa2S	MRR of Average	Difference	Question
1	474	0.000	0.717	-0.717	*Who first broke the sound barrier?*
2	859	0.000	0.606	-0.606	*Where is Rider College?*
3	614	0.000	0.602	-0.602	*Who wrote the book, "Huckleberry Finn"?*
4	270	0.000	0.601	-0.601	*Where is the Orinoco?*
5	363	0.000	0.589	-0.589	*What is the capital of Haiti?*
6	698	0.000	0.588	-0.588	*Where is Ocho Rios?*
7	495	0.000	0.579	-0.579	*When did Aldous Huxley write, "Brave New World"?*
8	727	0.000	0.578	-0.578	*Where is Procter & Gamble based in the U.S.?*
9	440	0.000	0.574	-0.574	*Where was Poe born?*
10	378	0.000	0.573	-0.573	*Who is the emperor of Japan?*

Table 5: Worst results and questions

ance density of the query term. Another problem of our system is that a correct answer cannot be consistently acquired; the phrase before and behind that is occasionally extracted. Thus, it was necessary to use linguistic information that cans more detailed extraction in the answer generation part in restricted length.

5 Summary

We described our TREC-9 QA processing system and discussed the result of our experimental retrieval searches. It was determined from an analysis of the data that the results of our initial search were roughly excellent result. However, we found that even if an initial search is successful, the correct answer could not always be correctly extracted. Our results suggested that the correct answer could be extracted roughly by a traditional information retrieval technique in the QA retrieval, but that natural language processing and the information extraction processing are indispensable for a complete extraction. We will examine the application of linguistic processing and information extraction to a QA retrieval technique using a key phrase within the question sentence in the future.

References

[1] S.E. Robertson, S. Walker, and M. Beaulieu, Okapi at TREC-7: automatic ad hoc, filtering, VLC and interactive, in *Proc. of the Seventh Text REtrieval Conference (TREC-7)*, NIST Special Publication 500-242, pp.253 - 264, 1999.

[2] T. Takaki, NTT DATA: Overview of system approach at TREC-8 ad hoc and question answering, in *Proc. of the Eighth Text REtrieval Conference (TREC-8)*, NIST Special Publication 500-246, pp.523 - 530, 2000.

[3] E. Voorhees, The TREC-8 Question Answering Track Report, in *Proc. of the Eighth Text REtrieval Conference (TREC-8)*, NIST Special Publication 500-246, pp.77-82, 2000.

Further Analysis of Whether Batch and User Evaluations Give the Same Results With a Question-Answering Task

William Hersh, Andrew Turpin, Lynetta Sacherek, Daniel Olson
Susan Price, Benjamin Chan, Dale Kraemer
Division of Medical Informatics & Outcomes Research
Oregon Health Sciences University
Portland, OR, USA

In the TREC-8 Interactive Track, our results indicated that the better performance obtained in batch searching evaluation do not translate into better performance by users in an instance recall task. This year we pursued this investigation further by performing the same experiments using the new question-answering task adopted in the TREC-9 Interactive Track. Our results once again show that better performance in batch searching evaluation does not translate into gains for real users.

A continuing unanswered question in information retrieval (IR) research is whether batch and user searching evaluations give the same results. We explored this question in the TREC-8 Interactive Track, where we found that the better results obtained in batch studies using the Okapi weighting scheme over the standard TFIDF approach did not accrue to real users for an instance recall task.[1] This work was limited by the small number of queries as well as the use of a single retrieval task, the recall of specific instances for a topic. Since the TREC-9 Interactive Track would be using a different task - question-answering - we decided to use the same research question again with this changed task. Although we would still have a small number of queries, it would provide another IR task to assess this research question.

As with the TREC-8 Interactive Track we performed three experiments. The first experiment was to identify an IR approach that achieved the best possible performance in the batch environment. In the second experiment, we used that best weighting measure as the "experimental" system to be compared with the "control" system using baseline TFIDF weighting. In the final experiment, we verified that the better batch performance of the experimental system held up with the new TREC-9 Interactive Track data.

Experiment 1 - Identifying the "best" weighting scheme

In TREC-8, the best weighting scheme was chosen by turning Interactive Track data from TREC-6 and TREC-7, which also used an instance recall task, into a test collection. All documents which had one or more instances were deemed relevant, and many runs using variants of TFIDF, Okapi, and pivoted normalization were used. The collection was that used by the instance recall task, the Financial Times 1991-1994 (FT91-94) from Disk 4 of the TREC CD-ROMs. The queries used were derived from the Description field of the topic. The Okapi weighting gave the best mean average precision (MAP), which was 83% over the standard TFIDF baseline.

All of our batch and user experiments used the MG retrieval system. [2] MG allows queries to be entered in either Boolean or ranked mode. If ranking is chosen, the ranking scheme can be varied according to the Q-expression notation introduced by Zobel and Moffat. [3] A Q-expression consists of eight letters written in three groups, each group separated by hyphens. For example, BB-ACB-BCA, is a valid Q-expression. The two triples describe how terms should contribute to the weight of a document and the weight of a query respectively. The first two letters of each triple define how a single term contributes to the document/query weight. The final letter of each triple describes the document/query length normalization scheme. The second character of the Q-expression details how term frequency should be treated in both the document and query weight, e.g., as inverse document/query frequencies. Finally, the first character determines how the four quantities (document term weight, query term weight, document normalization, and query normalization) are combined to give a similarity measure between any given document and query. To determine the exact meaning of each character, the five tables appearing in the Zobel and Moffat paper must be consulted. [3] Each character provides an index into the appropriate table for the character in that position.

Although the Q-expressions permit thousands of possible permutations to be expressed, several generalizations can be made. Q-expressions starting with a B use the cosine measure for combining weights, while those starting with an A do not divide the similarity measure by document or query normalization factors. A B in the second position indicates that the natural logarithm of one plus the number of documents divided by term frequency is used as a term' s weight, while a D in this position indicates that the natural logarithm of one plus the maximum term frequency divided by term frequency is used. A C in the fourth position indicates a cosine-measure-based term frequency treatment, while an F in this position indicates Okapi-style usage. [4] Varying the fifth character alters the document length normalization scheme. Letters greater than H use pivoted normalization. [5]

Methods

For the question-answering task of the TREC-9 Interactive Track, we had no prior Interactive Track data to use. Instead, we used almost all queries from the ad hoc collection dating back to TREC-2 (051-450) along with 20 prior instance recall queries (from the past three years of the Interactive Track) and the 200 queries from the TREC-8 question-answering track. For the latter, we deemed any document which had an answer string as relevant. Mean average precision was calculated using the trec_eval program.

Results

While the version of Okapi used in TREC-8 (AB-BFD-BAA) did better on instance recall queries from past Interactive Track experiments (using the FT91-94 collection), it did not perform as well on the other query-collection sets. The weighting scheme giving the best results over all of the query sets was a version of Okapi that employed pivoted normalization (AE-BFM-ABA) as shown in Table 1.

The new best Okapi weighting calculates the similarity between a document and query as

$$\sum_{t\in T_{q,d}} f_{q,t} \times \ln\left(\frac{N-f_t}{f_t}\right) \times \frac{f_{d,t}}{\left(f_{d,t} + \dfrac{W_d}{av(W_d)}\right)}$$

where

W_d	$(1-slope)+slope \times f_d$
$av(W_d)$	average W_d over all documents
N	number of documents in the collection
f_t	number of documents containing term *t*
$f_{q,t}$	frequency of the term in the query
$f_{d,t}$	frequency of the term in the document
$T_{q,d} =$	Set of terms both in *q* and *d*

Table 1 - Batch results for ad hoc, instance recall, and question-answering tasks using cosine TFIDF, Okapi weighting, and Okapi + pivoted normalization weighting.

Query set	Collection	Cosine	Okapi (% improvement)	Okapi + Pivoted normalization (% improvement)
303i-446i	FT91-94	0.2281	0.3753 (+65)	0.3268 (+43)
051-200	Disks 1&2	0.1139	0.1063 (-7)	0.1682 (+48)
202-250	Disks 2&3	0.1033	0.1153 (+12)	0.1498 (+45)
351-450	Disks 4&5 minus CR	0.1293	0.1771 (+37)	0.1825 (+41)
001qa-200qa	Disks 4&5 minus CR	0.0360	0.0657 (+83)	0.0760 (+111)
Average improvement			(+38)	(+58)

Table 2 - Mean average precision for various slopes, with 0.6 obtaining the best results.

Slope	Mean Average Precision
0.550	0.0740
0.575	0.0781
0.600	0.0782
0.650	0.0780
0.675	0.0776

and the sum is over all terms that occur both in the query and document.

As this new Okapi approach uses pivoted normalization, we needed to determine the best slope. As shown in Table 2, a slope of 0.6 was determined to be best.

The baseline TFIDF Q-expression was the same as for TREC-8 (BB-ACB-BAA), which calculates similarity between a document and the query as

$$\sum_{t \in T_{q,d}} \frac{\left(1+\ln f_{d,t}\right) \times \ln\left(1+\frac{N}{f_t}\right)}{\sqrt{\sum_{t \in doc}\left(1+\ln f_{d,t}\right)^2}}$$

where

N	number of documents in the collection
f_t	number of documents containing term t
$f_{q,t}$	frequency of the term in the query
$f_{d,t}$	frequency of the term in the document
$T_{q,d} =$	Set of terms both in q and d

Experiment 2 - Interactive retrieval experiments

Based on the results from Experiment 1, the goal of our interactive experiment was to assess whether the AE-BFM-ABA weighting scheme provided benefits to real users in the TREC interactive setting. The OHSU TREC-9 Interactive Track experiments were carried out according to the consensus protocol developed for the track. We used all of the instructions, worksheets, and questionnaires developed by consensus, augmented with some additional instruments, such as tests of cognitive abilities and a validated user interface questionnaire.

Methods

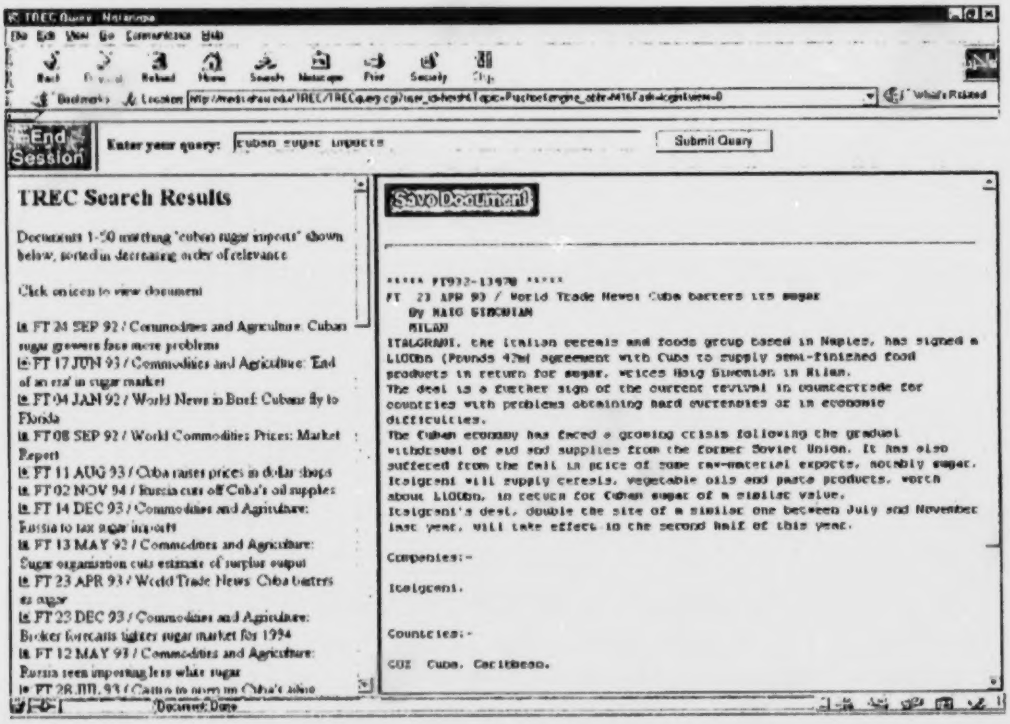
As noted above, the TREC-9 Interactive Track used a question-answering task. A set of eight questions was developed (see Table 3). Questions were of two types. The first type required users to find a small number of instances for a topic, e.g., the number of parks in the United States containing redwood trees. The second type required users to select the correct answer from two given, e.g., which country had a larger population, Denmark or Norway. Searchers from all sites were asked to answer the questions by searching, recording the answer, and recording all documents that contributed to the answer. Assessors at NIST scored each answer as being completely correct, partially correct, or not correct, with the documents saved by the user being judged as completely answering the question, partially answering the question, or not answering the question. For our analysis, a question was deemed correct if the assessor found the answer completely correct and the answer was supported by all documents saved by the user.

The collection used for these experiments was the same as that used by the question-answering track, consisting of Disks 4 and 5 (minus the Federal Register) of the TREC CD-ROM collection.

Both the baseline and the Okapi plus pivoted normalization systems used the same Web-based, natural language interface shown in Figure 1. MG was run on a Sun Enterprise 250 with 1 gigabyte of RAM running the Solaris 2.7 operating system. The user interface accessed MG via CGI scripts which contained JavaScript code for designating the appropriate weighting scheme and logging search strategies, documents viewed (title displayed to user), and documents seen (all of document displayed by user). Searchers accessed each system with either a Windows 95 PC or an Apple PowerMac, running Internet Explorer or Netscape Navigator.

Table 3 - Questions for interactive question-answering task.

1. What are the names of three US national parks where one can find redwoods?
2. Identify a site of Roman ruins in present day France?
3. Name four films in which Orson Welles actually appeared.
4. Name 3 countries that imported Cuban sugar during the period of time covered by the collection.
5. Which children's TV program was on the air longer, the original Mickey Mouse Club or the original Howdy Doody Show?
6. Which painting did Edvard Munch complete first, "Vampire" or "Puberty"?
7. Which was the last dynasty of China: Qing or Ming?
8. Is Denmark larger or smaller in population than Norway ?



Enter your query: cuban sugar imports

TREC Search Results

Documents 1-50 matching "cuban sugar imports" shown below, sorted in decreasing order of relevance

Click on icon to view document

- FT 24 SEP 92 / Commodities and Agriculture: Cuban sugar growers face more problems
- FT 17 JUN 93 / Commodities and Agriculture: End of an era in sugar market
- FT 04 JAN 92 / World News in Brief: Cubans fly to Florida
- FT 08 SEP 92 / World Commodities Prices: Market Report
- FT 11 AUG 93 / Cuba raises prices in dollar shops
- FT 02 NOV 94 / Russia cuts off Cuba's oil supplies
- FT 14 DEC 93 / Commodities and Agriculture: Russia to tax sugar imports
- FT 13 MAY 92 / Commodities and Agriculture: Sugar organisation cuts estimate of surplus output
- FT 23 APR 93 / World Trade News: Cuba barters its sugar
- FT 23 DEC 93 / Commodities and Agriculture: Broker forecasts tighter sugar market for 1994
- FT 12 MAY 93 / Commodities and Agriculture: Russia seen importing less white sugar
- FT 28 JUL 93 / Castro to open up Cuba's

```
***** FT932-13970 *****
FT  23 APR 93 / World Trade News: Cuba barters its sugar
   By HAIG SIMONIAN
   MILAN
ITALGRANI, the Italian cereals and foods group based in Naples, has signed a
L100bn (Pounds 42m) agreement with Cuba to supply semi-finished food
products in return for sugar, writes Haig Simonian in Milan.
The deal is a further sign of the current revival in countertrade for
countries with problems obtaining hard currencies or in economic
difficulties.
The Cuban economy has faced a growing crisis following the gradual
withdrawal of aid and supplies from the former Soviet Union. It has also
suffered from the fall in price of some raw-material exports, notably sugar.
Italgrani will supply cereals, vegetable oils and pasta products, worth
about L100bn, in return for Cuban sugar of a similar value.
Italgrani's deal, double the size of a similar one between July and November
last year, will take effect in the second half of this year.

Companies:-

Italgrani.


Countries:-

CUZ  Cuba, Caribbean.
```

Figure 1 - Searching interface for baseline and Okapi weighting systems.

Subjects were recruited by advertising over several librarian-oriented listservs in the Pacific Northwest. The advertisement explicitly stated that we sought information professionals with a library degree and that they would be paid a modest honorarium for their participation. We also recruited graduate students from the Master of Science in Medical Informatics Program at Oregon Health Sciences University (OHSU). They had a variety of backgrounds, from being a physician or other health care professionals to having completed only undergraduate studies.

The experiments took place in a computer lab. Each session took two hours, broken into three parts, separated by short breaks: personal data and attributes collection, searching with one system, and searching with the other system. The entire process included the following steps:

1. Orientation to experiment (10 minutes)
2. Administration of Pre-Search Questionnaire (10 minutes)
3. Orientation to searching session and retrieval system (10 minutes)
4. Practice search (10 minutes)
5. Short Break (5 minutes)
6. Searching on first 4 topics with assigned system (30 minutes)
7. Short break (10 minutes)
8. Searching on second 4 topics with assigned system (30 minutes)
9. Administration of Exit Questionnaire (5 minutes)

BEST COPY AVAILABLE

Each participant was assigned to search four questions in a block with one system followed by four questions with the other system. A pseudo-random approach was used to insure that all topic and system order effects were nullified. (A series of random orders of topics with subject by treatment blocks were generated (for balance) and used to assign topics.)

Per the consensus protocol, each participant was allowed five minutes per question. Participants were instructed to write their answer on the searcher worksheet and save all documents that supported their answers (either by using the "save" function of the system or writing its document identifier downon the searcher worksheet). The results of several participants had to be discarded for failing to follow these instructions.

The exit questionnaire was augmented from the consensus protocol to include the Questionnaire for User Interface Satisfaction (QUIS) 5.0 instrument [6]. QUIS provides a score from 0 (poor) to 9 (excellent) on a variety of user factors, with the overall score determined by averaging responses to each item. QUIS was given only at the end as a measure of overall user interface satisfaction since the interfaces for the two systems were identical.

For statistical analysis, we fit a series of mixed-model analysis of variance models and covariance models to the data. Mixed models allow both fixed effects (system and questions) and random effects (subjects) to be fit in one model. Given the binary outcome (correct or not correct), we fit a logistic model using a generalized linear model approach. We fit the model using SAS® Version 8.0 MACRO GLIMMIX, which uses an iteratively reweighted likelihood approach to fit these models. [7]

Our base model included systems (TFIDF and Okapi plus pivoted normalization) and questions. In addition to system and questions, since each subject answered all questions, we included subject in the model as a random intercept term. We also allowed a separate variance structure for each subject using the mixed model approach. In additional analyses we also added one of 11 covariates to the analysis of variance model (one covariate per analysis) to determine if the covariate made a significant contribution to the model with systems and questions. The covariates represented the factors measured in the various questionnaires and were each based on a Likert scale with values of one to five used as scale variables. The covariates and the variables they represent are listed in Table 4.

Table 4 - Covariates and the variables they represented.

Covariate	Definition
Familiar	User familiar with topic of question
Certainty	User certainty of answer
Easy Start	Easy to get started on question
Easy To Do	Question easy to answer
Satisfied	User satisfied system helped answer question
Time Adequate	Time was adequate to answer question
Terms	Number of unique terms used in all searchers for question
Cycles	Number of search cycles for question
Viewed	Number of document surrogates viewed for question
Seen	Number of documents for which full-text viewed for question
Saved	Number of documents saved as answering questions

Results

A total of 25 individuals followed instructions well enough for their data to be included in the analysis. Although a "pure" statistical analysis would only include the 16 subjects who have been balanced for query and system order, we have included the results from all 25 searchers in this initial analysis. The make-up of the participants was 18 librarians and seven others who were graduate students or research assistants. The average age of the librarians was 38.6 years. All but three were female. The average age of the remaining subjects was 34.4 years, with four males and three females.

The Pre-Search Questionnaire showed this was a group with a great deal of searching experience. Most had been searching for over half of their adult life. Virtually all reported high experience with point-and-click user interfaces, on-line library card catalogs, on-line searching, and Web searching. All indicated they frequently conducted searches and enjoyed doing it. Because of the heterogeneity of this data, no further analysis of this per-user data was performed. We instead focused our analysis on attributes measured on a per-question basis as described below.

The rate of correctness varied widely across the questions. Table 5 shows the results for each question based on the correctness criteria defined above, with results shown for all participating groups and OHSU searchers only. For the statistical analysis, we deleted two of the eight questions (numbers 3 and 8) because all searchers gave the same answer. Including a question in the analysis for which all subjects have the same answer, either correct or incorrect, causes problems for the iterative statistical algorithm. No subject answered either of the two deleted questions correctly. No question was answered correctly by all subjects. For OHSU searchers, the differences across questions was statistically significant using a Chi-square test ($p < .0001$). The rate of correctness did not vary, however, across systems. As shown in Table 6, it was virtually identical for the two retrieval systems. There was no statistically significant difference between systems.

Table 5 - Results for each question for all participants and OHSU participants only.

	All Groups			OHSU only		
Question	Incorrect	Correct	% Correct	Incorrect	Correct	% Correct
1	99	8	7.5%	21	4	16.0%
2	80	18	18.4%	20	5	20.0%
3	103	3	2.8%	25	0	0.0%
4	77	29	27.4%	10	15	60.0%
5	41	65	61.3%	5	20	80.0%
6	59	41	41.0%	6	19	76.0%
7	28	77	73.3%	4	21	84.0%
8	92	9	8.9%	25	0	0.0%
Total	579	250	30.2%	116	84	42.0%

Table 6 - Results for each question per system.

	TFIDF			Okapi + Pivoted Normalization		
Question	Searches	#Correct	%Correct	Searches	#Correct	%Correct
1	13	3	23.1%	12	1	8.3%
2	11	0	0.0%	14	5	35.7%
3	13	0	0.0%	12	0	0.0%
4	12	7	58.3%	13	8	61.5%
5	12	9	75.0%	13	11	84.6%
6	15	13	86.7%	10	6	60.0%
7	13	11	84.6%	12	10	83.3%
8	11	0	0.0%	14	0	0.0%
Total	100	43	43.0%	100	41	41.0%

The results of the analyses of covariance are shown in Table 7. None of the variables assessed were statistically significant by system and all were statistically significant by question. The latter, of course, represented the large variation in rate of correctness per question. There was a significant association with the following covariates: certainty, easy to do, satisfied, time adequate, seen, and saved. For satisfied and time adequate, the inclusion of the covariate resulted in a change in the p-value for questions. While this p-value was still significant at a 5% level, the p-values were much closer to the 5% than without the covariate. This suggests that the covariate was explaining some of the variation formerly explained by questions alone. There did not appear to a meaningful association between the other six covariates and the likelihood of being correct.

Experiment 3 - Verifying "best" weighting scheme

The final experiment was to determine whether the question-answering data for the TREC-9 Interactive Track gave better results in batch searching evaluation. This would allow us to determine whether the user evaluation in Experiment 2 gave the same or different results than batch searching experiments.

Table 7 - Summary of p-values for base analysis of variance model and model with each potential covariate added to model individually.

Covariate	System	Questions	Covariate
None	0.73	<0.0001	N/A
Familiar	0.76	<0.0001	0.70
Certainty	0.92	<0.0001	<0.0001
Easy Start	0.82	<0.0001	0.18
Easy To Do	0.82	0.0021	<0.0001
Satisfied	0.76	0.034	<0.0001
Time Adequate	0.88	0.030	<0.0001
Terms	0.98	<0.0001	0.096
Cycles	0.94	<0.0001	0.44
Viewed	0.86	<0.0001	0.13
Seen	0.59	<0.0001	0.0417
Saved	0.23	<0.0001	<0.0001

Methods

For this experiment, we developed a test collection consisting of the collection used for Experiment 2, queries derived from the question statement, and relevant judgments derived by designating those determined to "support" the answer by NIST assessors. In their judgment of the results, the assessors selected the correct answers as well as listing which documents provided "supporting" evidence for those answers. We assumed these documents were relevant and used them in our batch experiments accordingly.

Results

As shown in Table 8, the Okapi (AE-BFM-ABA) weighting provided improved MAP over TFIDF for all but on query and by an overall average of 31.5%. This was similar to our TREC-8 Interactive Track experiments, where batch results showed improved performance for the better weighting scheme that did not occur with user experiments.

Conclusions

Our TREC-9 Interactive Track results paralleled our TREC-8 results, i.e., performance enhancement that occurred in batch evaluation studies was not associated with performance of real users. As with our TREC-8 Interactive Track study, this one had limitations as well. Like past experiments, the number of queries and users was small. Recent research suggests that evaluation measures are unstable when less than 25 or 50 queries are used in an evaluation, at least in the batch setting. [8] Nonetheless, there is some significance to the fact that comparable results have been obtained with two different retrieval tasks even with the small number of queries and users.

A number of factors were assessed to determine their effect on the rate of correctness, but the large variation in question correctness overwhelmed any differences in effects of the factors.

The next step in our research will be an investigation to determine why gains in batch evaluation performance do not occur in real user studies. There are really two possibilities: either real users do not get the kind of improved recall and precision seen in batch studies with the queries that they enter or they do get better recall and precision in their searches but it does not translate into better user performance with the specific task. We will assess this by calculating recall and precision on the actual queries entered by users to determine whether systems using Okapi weighting provide benefit to them.

Table 8 - Batch searching results.

Question	TFIDF	Okapi + Pivoted Normalization	% improvement
1	0.1352	0.0635	-53.0%
2	0.0508	0.0605	19.1%
3	0.1557	0.3000	92.7%
4	0.1515	0.1778	17.4%
5	0.5167	0.6823	32.0%
6	0.7576	1.0000	32.0%
7	0.3860	0.5425	40.5%
8	0.0034	0.0088	158.8%
Mean	0.2696	0.3544	31.5%

Acknowledgements

This study was supported in part by Grant LM06311 of the U.S. National Library of Medicine.

References

[1] Hersh W, et al. *Do batch and user evaluations give the same results?*, in *Proceedings of the 23rd Annual International ACM Special Interest Group in Information Retrieval.* 2000. Athens, Greece: ACM Press, 17-24.

[2] Witten I, Moffat A, and Bell T, *Managing Gigabytes - Compressing and Indexing Documents and Images*. 1994, New York: Van Nostrand Reinhold.

[3] Zobel J and Moffat A, *Exploring the similarity space.* SIGIR Forum, 1998. 32: 18-34.

[4] Robertson S and Walker S. *Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval*, in *Proceedings of the 17th Annual International ACM Special Interest Group in Information Retrieval.* 1994. Dublin: Springer-Verlag, 232-41.

[5] Singhal A, Buckley C, and Mitra M. *Pivoted document length normalization*, in *Proceedings of the 19th Annual International ACM Special Interest Group in Information Retrieval.* 1996. Zurich, Switzerland: ACM Press, 21-9.

[6] Chin J, Diehl V, and Norman K. *Development of an instrument measuring user satisfaction of the human-computer interface*, in *Proceedings of CHI '88 - Human Factors in Computing Systems*. 1988. New York: ACM Press, 213-8.

[7] Wolfinger R and O'Connell M, *Generalized linear mixed models: a pseudo-likelihood approach.* Journal of Statistical Computation and Simulation, 1993. 48: 233-43.

[8] Buckley C and Voorhees E. *Evaluating evaluation measure stability*, in *Proceedings of the 23rd Annual International ACM Special Interest Group in Information Retrieval.* 2000. Athens, Greece: ACM, 33-40.

TREC-9 Cross Language, Web and Question-Answering Track Experiments using PIRCS

K.L. Kwok, L. Grunfeld, N. Dinstl and M. Chan
Computer Science Department, Queens College, CUNY
Flushing, NY 11367

Abstract

In TREC-9, we participated in the English-Chinese Cross Language, 10GB Web data ad-hoc retrieval as well as the Question-Answering tracks, all using automatic procedures. All these tracks were new for us.

For Cross Language track, we made use of two techniques of query translation: MT software and bilingual wordlist lookup with disambiguation. The retrieval lists from them were then combined as our submitted results. One submitted run used wordlist translation only. All cross language runs make use of the previous TREC Chinese collection for enrichment. One MT run also employs pre-translation query expansion using TREC English collections. We also submitted a monolingual run without collection enrichment. Evaluation shows that English-Chinese crosslingual retrieval using only wordlist query translation can achieve about 70-75% of monolingual average precision, and combination with MT query translation further brings this effectiveness to 80-85% of monolingual. Results are well-above median.

Our PIRCS system was upgraded to handle the 10GB Web track data. Retrieval procedures were similar to those of the previous ad-hoc experiments. Results are well-above median.

In the Question-Answering track, we analyzed questions into a few categories (like 'who', 'where', 'when', etc.) and used simple heuristics to weight and rank sentences in retrieved documents that may contain answers to the questions. We used both the NIST-supplied retrieval list and our own. Results are also well-above median.

Two runs were also submitted for the Adaptive Filtering track. These were done using old programs without training because we ran out of time. Results were predictably unsatisfactory.

1 Introduction

By some coincidence, all the tasks that we participated in TREC-9 were to us either new or involve new processing of collections. We managed to complete three of the four tasks that we initially targeted with very good results. These are cross language information retrieval (Section 2), the 10GB web data retrieval (Section 3) and the question-answering track (Section 4). The adaptive filtering track (Section 5) was done with little preparation and the result was poor. Section 6 has the conclusions.

2 English-Chinese Cross Language IR

The aim of the task is to retrieve from a Chinese collection documents relevant to queries given in English. The collection consists of about 210 MB of text from three Hong Kong newspapers. Twenty-five queries (#55 to #79) were provided in both English and Chinese. We employed the query translation approach to CLIR by translating the English queries and retrieve in monolingual Chinese. The task is complicated by the fact that the Chinese collection is encoded in BIG5 while our translation resources are mainly GB-code oriented. Since no translation methodology is perfect, we rely on multiple (two) translation methods and retrieval combination technique to lessen wrong or null translations consequences and to provide better results than using one single methodology.

2.1 Query Translation Methodologies

The 25 English topics were first pre-processed by our system to remove some non-content introductory phrases. In addition, sentences that contain negation such as 'not relevant', 'irrelevant', 'non-relevant' are also discarded. We noticed that many narrative sections actually contain only one such sentence, and hence such topics would effectively contain only a title and a descriptive section only. The 25 queries have an average of 9.44 English terms.

The first translation method is based on commercial MT software. Such PC software for English-Chinese are quite common nowadays, costing between scores to about a thousand dollars for a single user license. We consider MT software as a poor man's way of gaining access to a bilingual dictionary with disambiguation technique built-in. For statistical IR, the output that counts is mainly the accuracy of content term translations; other factors such as style, word order, readability, etc. are not important. We tested several packages and finally decided on one called HuaJian (http://www.altlan.com) from Mainland China. It performs very well for the 54 long and short topics and 160MB Chinese collection of TREC 5&6. For example, its untouched translation output attains over 80% of monolingual results. This is used for TREC-9. An example of its quality is shown later in Section 2.2.

A second approach we used for translation is based on automatic dictionary lookup. Most bilingual dictionaries on the web or sold commercially are designed for consultation only. Downloadable dictionaries that can be accessed by program are rare. The LDC (Linguistics Data Consortium) however has compiled two fairly comprehensive English/Chinese wordlists of about 120K in size each, and are available for research purposes (http://www.morph.ldc.edu/Projects/Chinese). One is for English to Chinese, and the other the reverse, and is reported to have similar content. We studied both [Kwok00] and finally decided that the Chinese-to-English version ldc2ce is much more useful for translation purposes because of its dictionary structure. Example entries of the ldc2ce wordlist are shown below:

1	人性	/human/
2	人类	/humanity/human race/mankind/
3	人权	/human rights/
4	人权观察	/Human Rights Watch (organization)/
5	人体	/human body/
6	风土人情	/local conditions (human and environmental)/
7	最惠国	/most-favored nation (trade status)/

It is seen that if a query has the word 'human', one can pick up several mappings that contain this English word in the explanations of lines 1-6. However, because of the wordlist structure, only one of them (line 1) has a precise translation – the other lines may have meaning (and their translation) being contaminated by the way 'human' is used in association with other words. Thus, we have a natural way of disambiguating these multiple translations. Moreover, if the word 'human' occurs as a phrase like 'human rights' in the query, one can also perform string matching in the explanations to pick up line 3 as the sole translation for the phrase instead of individual single word translations. Phrase translations generally are unambiguous and play an important role [BaCr97] for accurate cross language retrieval. Thus, the Chinese-English wordlist can be regarded as both a word and phrase dictionary.

Even with the above considerations, many single words still remain with a large number of mappings. To further disambiguate them, we rely on the retrieval corpus term statistics to help weed down this number. The hypothesis is that the larger the term's occurrence in a corpus, the higher the probability that the term is a good translation. Thus, for a set of candidate translations of an English word, we keep only the top n most frequent (after ignoring stopwords). However, choosing the threshold n is problematic. Too small a number risks leaving out a correct translation, while too large a number means keeping too much noise. Interestingly, in [Pirkola 98] a method of weighting translations is introduced that allows one can to keep a larger number of translations without seeing the effect of noise. This method is to regard the candidate translations as a synonym set with each term having a collection frequency equal to the sum of the set. Thus, low occurrence frequency terms that are included would not unduly influence the resultant query. Our experiments allow a maximum of six candidate translations to be kept, and this has worked well with the TREC 5&6 Chinese collections in a cross language retrieval environment.

The ldc2ce wordlist discussed earlier is GB-coded, and historically it may have been derived from Mainland China documents. Since our target retrieval collection is in BIG5 and derived from newspapers in Hong Kong, there may be a mismatch in term usage. In the LDC website there is also an available parallel corpus whose content is Hong Kong government laws. Buried in the documents there are many content words or phrases that are followed with translations in parenthesis. We mined some 6000 such translation pairs, converted to GB code, and added to the ld2ce wordlist. This is our resultant translation wordlist.

For the 25 queries, 6 phrases (total 10 with repeats) are extracted. An example query translated via our wordlist is shown below. Numeric values show how many mappings are found for each English word (maximum 5 in this example). They are delimited by ^ as a group. For example, both 'air' and 'pollution' (first two words) are mapped into three Chinese terms. One phrase translation of 'government organizations' is correctly picked up. The word 'auto' was assigned two Chinese terms with different senses 'automobile' and 'automatic'. The HuaJian MT translation is also shown, and it is seen that it picks up 'air pollution' correctly but misses out the 'automobile' sense of 'auto'. Overall, both translations are quite adequate for CLIR.

Query #CH75 Original English

Air pollution in China .

China's efforts in reducing air pollution, including the government organizations involved and their effectiveness in dealing with air pollution in China.

All types of air pollution are relevant, including industrial, auto emissions, and air pollution from private sources. that reports a reduction or an increase in air pollution in China is considered relevant.

Query #CH75 Translation using ldc2ce

^0.33 空气 0.33 样子 0.33 调 ^^0.33 污
0.33 败坏 0.33 弄脏 ^
IN ^0.50 中国 0.50 中华 ^.
^0.50 中国 0.50 中华 ^^0.33 功夫 0.33 承诺
0.33 工夫 ^ IN
^1.0 减轻体重法^^0.33 空气 0.33 样子 0.33
调 ^^0.33 污 0.33 败坏 0.33 弄脏 ^^0.50
在内 0.50 包含 ^
THE ^1.0 政府机构^ ^1.0 亲^
AND ^0.50 其 0.50 亓 ^^0.33 有效 0.33 有力
0.33 效用 ^
IN ^0.25 处理 0.25 往来 0.25 生意 0.25 往还 ^
WITH
^0.33 空气 0.33 样子 0.33 调 ^^0.33 污
0.33 败坏 0.33 弄脏 ^
IN ^0.50 中国 0.50 中华 ^.
ALL ^0.25 类型 0.25 样式 0.25 试样 0.25 种 ^
OF
^0.33 空气 0.33 样子 0.33 调 ^^0.33 污
0.33 败坏 0.33 弄脏 ^ ARE
^0.50 有关 0.50 相应 ^^0.50 在内 0.50 包含 ^
^1.0 产 ^^0.50 汽车 0.50 自动 ^ ^1.0 排放^
AND
^0.33 空气 0.33 样子 0.33 调 ^^0.33 污
0.33 败坏 0.33 弄脏 ^
FROM
^0.50 私营 0.50 私人 ^^0.17 引起 0.17 来源
0.17 源头 0.17 本源 0.17 情报处 0.17 源点 ^.
THAT ^0.50 报告 0.50 汇报 ^^0.50 降价 0.50
裁减军备 ^
OR AN ^0.20 提高 0.20 增长 0.20 增添 0.20
益 0.20 茁 ^
IN ^0.33 空气 0.33 样子 0.33 调 ^^0.33 污
0.33 败坏 0.33 弄脏 ^
IN ^0.50 中国 0.50 中华 ^
IS ^0.50 考虑过 0.50 被尊重 ^^0.50 有关
0.50 相应 ^.

Query #CH75 Translation using HuaJian MT

在中国的空气污染…
中国努力在 方面 降低 空气污染，包 括政府
组织 包含和他们的效率
在在中国处理空气污染方面。
全部种空气污染是相应的 ， 包括工业国，
自动发射物和空气污染从 私 人资本。那报告一
减少或 者在在中国的空气污染方面的一次增加
被认为相应。

2.2 Query Processing

Each English query was translated into GB-coded Chinese either by HuaJian MT or by our dictionary process. They were then converted into BIG5 for retrieval by a program developed in house that has accuracy similar to the NJSTAR Communicator (http://www.njstar.com). The GB version is also retained to select documents from the TREC 5&6 Chinese GB-encoded collection for the purpose of collection enrichment described in Section 2.4. These selected documents were later converted into BIG5.

2.3 Document Processing

Since the collection is BIG5 encoded, we have modified our document processing programs to support this new coding. Because the queries will be obtained via translation, we also decided to use the translation wordlist as part of our segmentation dictionary to insure correct matching between query and document terms. However, only short words of four or less characters are kept. Our final segmentation dictionary size is about 100K. This is in contrast to our previous work on Chinese retrieval where we derived our segmentation dictionary of about 43K in size from the collection itself. We also follow our tradition to truncate long documents into sub-documents of about 550 characters in size ending on a paragraph. There were 127,938 documents producing a total of 211,536 sub-documents. The master dictionary has 102,156 unique terms. After stopword removal based on a threshold of 20,000, it is reduced to 53,462 terms for retrieval.

2.4 Retrieval Methodologies

After query translation is done, retrieval will be monolingual ad-hoc. However, many techniques can be used to improve retrieval accuracy. Based on experience with the TREC 5&6 Chinese collection used for cross language retrieval, we adopted the following procedures:

Pre-translation query expansion:
This means using the English queries to do retrieval on an English collection and employ pseudo-relevance feedback to expand the queries with English terms. This often can bring highly related terms and more focus on the query topic for later translation. We used this expansion with 15 terms only for queries to be translated via MT. For dictionary translation, we are more cautious as the new expanded terms may bring more noise than signal after translation.

Pseudo-relevance feedback:
This is sometimes known as post-translation query expansion in a cross language retrieval setting. The idea is to use the documents resulting from a first stage retrieval to define the domain of the query and add more Chinese terms to it. This can often lead to substantial improvements of 10 to 30%. Our PIRCS system uses this 2-stage retrieval as a default. We have employed a

standard of 40 top documents for feedback and 70 terms for query expansion.

Collection enrichment:

Pseudo-relevance feedback works only if the first stage retrieval results in a document list that is rich in relevant or highly-related documents. Collection enrichment is the technique of adding an external collection to the target collection in order to improve the probability of acquiring more relevant documents in this first-stage retrieval. The only available Chinese collections we have for this purpose are those of TREC 5&6. However, the latter collection is in GB coding different from the target which is in BIG5. Thus code conversion is necessary. Moreover, the collections are from different years, and have cultural differences (the target collection is from Hong Kong while the enrichment collection is from Mainland China). Thus there is a risk that the procedure may not work.

We are cautious about pre-translation expansion and collection enrichment and only used the procedure for selected runs discussed in the next section.

2.5 Results and Discussion

We submitted one monolingual retrieval pir0Xori as our basis, and three CLIR runs named: pir0Xdin, pir0Xhnd and pir0XHxD. Our convention for pir0X means PIRCS for year 2000 crosslingual experiments, and the last 3 characters differentiate the runs: **'ori'** is the original query monolingual, **'din'** (also referred to as ldc6n) uses our enhanced ldc wordlist with collection enrichment, **'hnd'** combines HuaJian MT (with enrichment) and wordlist without enrichment, and **'HxD'** combines MT with pre-translation expansion and wordlist translation – all with enrichment.

	Rel.retr	Avg.Pre	P@10	P@20	P@30
* ori	616 %	.285 %	.292 %	.236 %	.225 %
hjx0	469 .76	.195 .68	.224 .77	.182 .77	.151 .67
hjx15	566 .92	.206 .72	.208 .71	.158 .67	.143 .63
ldc6	568 .92	.196 .69	.220 .75	.192 .81	.176 .78
orn	613 1.0	.297 1.04	.276 .95	.252 1.07	.231 1.03
hjx0n	469 .76	.223 .78	.252 .86	.184 .78	.153 .68
hjx15n	563 .91	.213 .75	.232 .79	.172 .73	.152 .67
* din	575 .93	.216 .76	.232 .79	.194 .82	.175 .78
hjd	509 .83	.221 .78	.236 .81	.196 .83	.169 .75
* hnd	507 .82	.240 .84	.252 .86	.206 .87	.179 .79
hndn	493 .80	.245 .86	.260 .89	.198 .84	.173 .77
*** HxD**	**568 .92**	**.245 .86**	**.260 .89**	**.188 .80**	**.169 .75**

Table 2.1: Summary of Monolingual & Crosslingual Results

Internally we had many more runs, consisting of single translation methods: hjx0 and hjx15 (HuaJian MT without and with pre-translation expansion of 15 terms), hjx0n and hjx15n (same as before but with collection enrichment), ldc6 (wordlist only retaining maximum of 6 alternative translation), ldc6n which is also named pir0Xdin (ldc6 with collection enrichment), 'hjd' combines hjx0 with ldc6, and 'hndn' combines hjx0n with ldc6n. In addition, we had another monolingual run using collection enrichment called 'orn'. As discussed in Section 2.4, we do not know if enrichment using vastly different collections will work or not, and submitted the 'ori' monolingual run to be cautious. These results are shown in Table 2.1, where the * rows are our official submissions. The 'ori' row result is used as the basis (indicated by %) for measuring the various crosslingual retrievals. All our runs are automatic without human intervention.

It is surprising that the basic HuaJian MT (hjx0 – 68% monolingual in Avg.Pre) does not perform as well as for the TREC 5&6 environment (over 80% of monolingual). The basic wordlist (ldc6) approach performs as expected: 69% of monolingual in Avg.Pre and quite comparable to hjx0, with an edge for ldc6 – especially in the number of relevants-retrieved which attains an impressive 92% of monolingual. This is possibly due to the allowable 6 alternatives for each English word to be translated, while the MT software necessarily gives only one unique outcome. When pre-translation query expansion is used with MT (hjx15), this relevants-retrieved deficit is removed, but precision at low n suffers. Average precision however improves over both hjx0 and ldc6.

When the first 4 rows are compared with the next corresponding 4 that use collection enrichment, it is seen that this technique brings in 3 to 11% improvement among different measures except for two cases: hjx15n vs hjx15 where the relevants-retrieved practically remains unchanged, and orn vs ori where the precision at 10 documents declines by 5%. Otherwise, results show that collection enrichment works in the majority of cases even with such disparate collections. In particular, the monolingual run orn attains a 4% improvement over our official submission ori in average precision. Again, MT approach (hjx0n) shows good precision values but comparatively low relevants-retrieved. When pre-translation expansion is employed (hjx15n), this value is restored, but precision suffers. The 'din' (same as ldc6n) wordlist run attains good recall and precision in comparison. With collection enrichment, these cross language results now attain over 75% of 'ori' monolingual.

The final 4 rows show different combination runs. Results supports the fact that MT and wordlist approach seem to complement each other well, bringing average precision to 84 to 86% of monolingual. Collection enrichment seems to be an important factor to bring good results, as the 'hjd' row shows that plain hjx0 combined with ldc6 do not perform much better than their singleton runs with enrichment ('hjx0n' or 'din') and attains only about 78% of monolingual. Overall, the best result appears to be our submitted run HxD which combines MT with pre-translation query expansion, and wordlist approach and both with collection enrichment. For fairer comparison, we should use 'orn' (monolingual with enrichment) as the basis. In this case, HxD still attains over 82% of monolingual in average precision, and 93% in relevants-retrieved.

The next Table 2.2 shows how our submitted runs compare with others. For example, pir0XHxD has 17 better, 3 equal to median, and 5 worse for the Avg.Pre measure. pir0Xhnd also has 20 queries better or equal to median, and 5 worse. Of the 5, 1 query in 'hnd' is worst while HxD has 1 best among 17 better than median.

	pir0Xori			pir0Xdin			pir0Xhnd			pir0XHxD		
	>	=	<	>	=	<	>	=	<	>	=	<
AvgPrec	17,2	1	7	18,2	0	6,2	19	1	5,1	17,1	3	5
RR@100	19,6	3	3	16,6	5	4,2	16,5	6	3,2	15,5	8	2
RR@1K	20,6	3	2	18,10	5	2,2	18,11	3	4,1	19,11	3	3

Table 2.2 : Crosslingual Results: Comparing Submitted Runs with Median

We like to emphasize that these blind experimental results were achieved using publicly attainable resources.

3 10-GB Web Track

We participated in the Web Track the first time. The 10 GB represents a 5-fold increase in size from previous collections and is a challenge for our PIRCS system. From the raw text, we removed all the HTML tags like hypertext links, IMG elements, BACKGROUND, COLOR, WIDTH, HEIGHT and similar attributes. Heading and paragraph alignment attributes were replaced by a UNIX new line character. Entity or character references were also replaced by printable ASCII characters. Badly formed entity or character references were deleted. In order to reduce the inherent web data noise, we removed any contiguous strings that were longer than 32 characters. The data also contain many web pages in foreign languages like Spanish, German etc.; they were kept and not removed. To parse the text, we downloaded a C program written by Stephen M. Orth (Sorth@oz.net) and enhanced the program to fit our specific task.

As usual for our PIRCS processing, we broke long documents into approximately 3000 byte (instead of 550 words) long sub-documents ending at paragraph boundaries. This resulted in about 2.6 million sub-documents. After removing words that have a document frequency of less than 3 and more than 180,000, the resultant dictionary has 463K unique terms after stemming and stopword removal.

As before, the TREC-9 Web Track topics has several sections: title, description and narrative. This year we submitted five runs. Four are content-only while the fifth one tries to make use of the link information. The four content-only runs are named pir0Wt1, pir0Wtd2, pir0Wttd and pir0Watd. The prefix convention pir0W represents PIRCS runs year 2000 Web track. The last three characters differentiate the runs: t1 uses the title section only, td2 makes use of both the title and description, ttd is a combination of the retrieval lists from t1 and td1 (another title and description run that was not submitted; it differs from td2 in that the latter adds term variety to the query based on mutual information measure), and atd is a combination of the retrieval lists from pir0Wa1 and pir0Wtd1. a1 means using all sections of a topic.

The title, title-description, and all-section queries have 2.22, 5.32, and 9.12 unique terms respectively averaged over 50 queries. Our link-based run is called pir0WTTD and will be discussed in Section 3.3 while the content-based runs are discussed in Section 3.2.

3.1 General Methodology

We follow our TREC-8 ad-hoc approach by using four methods successively to produce a final retrieval list. These four methods [KwCh98] are: 1) average within-document term frequency to weight short query terms (avtf query term weighting); 2) variable high frequency Zipfian threshold dependent on query size; 3) collection enrichment to improve initial stage output relevant density; and 4) for td2 run only, enhancing term variety in raw queries by adding highly associated terms based on initial retrieval. For collection enrichment, we form a miscellaneous collection by retrieving the top 200 documents from the Question-Answering Track documents. This miscellaneous collection is used to enrich the top-ranked set of the initial stage retrieval. Second stage retrieval employs 25 top documents and 60 terms for pseudo-relevance feedback (long a1, and medium td queries). For short queries (t1) only 30 terms are added. Additionally, we use retrieval list combination to help improve effectiveness. The coefficients of combination are learnt from past results.

3.2 Content-based Retrieval

Our TREC-9 results are summarized in Table 3.1 and their nomenclature has been described previously. The title-description run is significantly better than that of title only run (td2 Avg.Pre 0.2164 vs. t1: avg. prec. 0.1750) -- an improvement of 24%. The lack-luster performance of the title run can be attributed to the fact that three of the queries have misspelled words. Query

464 ("nativityscenes"), query 487 ("angioplast7") and query 463 ("tartin") produce zero-length queries in our system (we do not perform spell-check and correction). In addition, query 456 ("is the world going to end") and 474 ("how e-mail bennefits businesses") also produce null queries (after stopword removal and stem conflation). They either contain high collection frequency terms like 'world', 'end', 'businesses' that are beyond our threshold and not retained in our dictionary or mis-spelling. We missed e-mail because it was not considered as a single word. Another query #475 ("the compostion of zirconium") also returns null retrieval list because of the mis-spelling "compostion" that has a legitimate but different meaning after stemming. Even though our initial retrieval list managed to return some documents, they are ranked far lower than the top 25 ranking. This leads to a 2nd retrieval with zero relevants. Another query (#473) has only 1 relevant document, and our system missed it also. Instead of returning an empty ranked list for null queries, our PIRCS engine generates randomly a list of one thousand documents in such circumstances. These lists do not help, and the Avg.Pre values are all zero. Totally we have seven queries with zero Avg.Pre. Adding the description to the query removes these difficulties.

Because the title only run (t1) is not good, its combination with td1 resulting in ttd does not give much improvement over td1. Also, when a1 is combined with td1 resulting in atd, its result is actually worse than a1 by itself. For these web data and questions, it appears that the title run is too poor for combination to work. The best of our submitted runs is pir0Watd. The average precision 0.2209 is 26% better that that of title only. It also has a relevant-retrieved at 1000 documents of 2011, which is about 77% of the pooled documents that have been judged relevant (2617).

Comparisons with the all-sites median average-precision, precision at 100 and 1000 documents are given in Table 3.2. Our content-only runs are well above the median. For example, pir0Watd has avenge-precision better or equal to median in 36 instances with 2 queries achieving the best, and is worse than the median in 14 cases. For title only, the number of queries with precision better, equal or worse than the median are: 32:4:14. Out of the 32 that are above median, 5 have the best value. The medians for title only and non-title-only run are evaluated separately.

Figures for precision at 100 and 1000 documents may be complicated to interpret since the total does not add up to 50 (the number of queries). The reason is that quite a few values are equal to zero. For example, the best, median and worst values for query 473 in title only run for precision at 100 document are all zero. Therefore, our score of zero means that our query 473 achieves the best, median and worst result all at the same time. But it is not better than the median nor it is worse than the median.

3.3 Link-based Retrieval

We tried one run, pir0WTTD, combining contend-based and link-based evidential information. The title-only run, pir0Wt1, retrieval list was used to perform the experiment using the link references in order to improve the retrieval ranking. We assume that a document referenced by many other documents in the output would indicate a higher relevance value compared to documents

					un-submitted				un-submitted					
	t1		**td1**		**td2**		**ttd**		**a1**		**atd**		**TTD**	
	value	% inc	value	% inc	value	% inc	value	% inc	value	% inc	value	% inc	value	% inc
Rel Retr	1518	0	2010	32	2010	32	2005	32	1915	26	**2011**	**32**	2005	32
Avg Prec	.1750	0	.2056	17	.2164	24	.2097	20	.2257	29	**.2209**	**26**	.1418	-19
Prec @ 10	.2180	0	.2960	36	.3020	39	.3180	46	.3320	52	**.2980**	**37**	.1800	-17
Prec @ 20	.1920	0	.2530	32	.2570	34	.2640	38	.2650	38	**.2750**	**43**	.1740	-9
Prec @ 30	.1773	0	.2307	30	.2393	35	.2327	31	.2360	33	**.2433**	**37**	.1680	-5
R-Precision	.1893	0	.2103	11	.2242	18	.2125	12	.2271	20	**.2275**	**20**	.1439	-24

Table 3.1: Automatic Web Track Results for the 50 Queries

	pir0Wt1	**pir0Wtd2**	**pir0Wttd**	**pir0Watd**	**pir0WTTD**
	> = <	> = <	> = <	> = <	> = <
Avg Prec	32,5 4 14	30,3 2 18	34,2 2 14	35,2 1 14	21 1 28
RR @ 100	29,11 15 6,10	30,9 12 8,3	31,10 12 7,2	37,11 8 5,3	23,6 12 15,2
RR @ 1K	32,24 10 8,7	33,23 14 3	35,26 13 2	36,27 12 2	35,26 13 2

Table 3.2: Web Track Results: Comparing Submitted Runs with Median

receiving less or no references, and that re-ranking the output based on this information will improve the result. We determined all incoming links for a document and calculated a link-value for that document (link-value = Ó(1 to 1000) (0.5 * log (1000 – source-rank)). A new rank was then calculated (new-rank = (old-rank + link-value)/2). The result was however disappointing. The table shows that this Avg.Pre value of 0.1418 (pir0WTTD) is considerably lower than the original content only result (pir0Wt1). Further investigation is necessary to determine the reason for the significantly lower results.

4 Query-Answering (QA) Track

4.1 Introduction

The QA Track involves 693 queries retrieving against a collection made up of: AP1-3, WSJ1-2, SJMN-3, FT-4, LA-5, and FBIS-5.

In [LeSJ96] Lewis and Sparck Jones contrast the promise of NLP retrieval systems to the basic statistical IR method. They observe, that while simple NLP strategies could improve text retrieval effectiveness, nevertheless statistical IR method 'has apparently picked some of the low-hanging fruit off the tree'. For example, statistical IR does not attempt word-sense disambiguation, yet 'when a document and a query match on several words, the individual matching words will have the same word sense'. Our QA system is constructed using the methods of classical IR, enhanced with some simple heuristics to pick off some more low-hanging fruit. Since our system lacks natural language understanding, the task is viewed as one of retrieving the best sentence, which is most likely to answer the query.

4.2 Components of our QA Approach

The simplest retrieval strategy seems to be 1) **coordinate matching,** a count of words in a document sentence matching the content words of the query. On top of this, we have added the following considerations:

2) **Stemming:** words are matched even if the are not exactly the same.
3) **Synonyms:** a hand created dictionary of some 300 terms. It contains unusual word forms, which are not handled well by stemming. Most of the entries were taken directly from Wordnet. More automatic use of Wordnet is contemplated for the future. There are four groups of synonym entries as shown in the sample Table 4.1.
4) **RSV:** the retrieval status values of the retrieval system. Given two sentences with the same score based on terms, preference is given to the one that is contained in a higher-ranking document.
5) **ICTF:** inverse collection term frequency gives more credit to less frequently occurring words. For practical reasons, the collection used to obtain the frequencies is the N top retrieved documents. This sometimes causes the system to misclassify the importance of a word. In the future we may want to use the statistics from the entire collection.
6) **Exact** important word: we give extra credit for words deemed important which must occur in the answer. At present, these are the superlatives: first, last, best, highest etc. However, one must be careful: 'best' is good but 'seventh best' is not.
7) **Proximity:** query words in close proximity in the sentence are likely to refer to the same concept as the query. This is currently done only, if all content query words are matched.

Description	Entry
Nationality	ROMAN ROME SPANISH SPAIN PORTUGUESE LUSITANIAN PORTUGAL SICILIAN SICILY FINNISH FINLAND SWEDISH SWEDEN DANISH DENMARK DANES BELGIAN BELGIUM LUXEMBOURGIAN LUXEMBOURG
Unusual Verb forms	KNEW KNOW KNOWN KNOWS LEND LENT LOST LOSE MISBECAME MISBECOME MISSPEND MISSPENT MISTOOK MISTAKE MISUNDERSTOOD MISUNDERSTAND MOLTEN MELT MOWN MOW MOWS
Noun synonyms	MALE MEN MAN FEMALE WOMEN WOMAN
Abbreviations	CAPT CAPTAIN UNITED STATES, US, USA, U.S. UNITED STATES OF AMERICA UNITED KINGDOM, UK U.K. UNITED NATIONS, UN U.N.

Table 4.1 Samples from Synonym Table

8) **Heading:** query words in the headline tag will receive credit if they do not occur in the sentence.
9) **Phrases:** if consecutive words in the query occur in consecutive order in the sentence.
10) **Caps:** capitalized query words.
11) **Quoted:** quoted query words.

A query-analyzer was built to recognize a number of specialized queries. 'Who', 'Where', 'What name' queries are processed by the capitalized answer module. 'When', 'How many', 'How much' and 'What number' are processed by the numerical answer module.

Name Answer Module: we included some simple heuristics to identify the following:

- Persons: Capitalized word not preceded by 'the'
- Places: Capitalized words preceded by 'on', 'in' and 'at'
- Capitalized words. When no other clues are available.

Numerical Answer Module:

- Units: there are classes of queries, which require units. Our system recognizes five types of units: length, area, time, currency and people. See Table 4.2 below
- Dates: There are some queries that have a date year in the question. This date must occur in the sentence or within the date tag.
- Numbers. When no other clues are available.

Type	Entry
Length	METER KM KILOMETER MILE KM CM FEET FT INCH FOOT MM MILIMETER
Area	SQ SQUARE ACRE
Time	MIN MINUTE DAY WEEK YEAR SECOND MONTH
Currency	DOLLAR $ YEN POUND
Population	PEOPLE INHABITANT POPULATION

Table 4.2 Units Recognized

These heuristics are of course not foolproof. For example we assume that a 'Where' question requires an upper case answer, which is not always the case. In particular the following queries have lower case answers:

227. Where does dew come from?
258. Where do lobsters like to live?
385. Where are zebras most likely found?

Selecting 50-byte answer from the top retrieved answer is quite a challenge. We used proximity to query words criterion for selection, and it misses many answers.

The contribution made by each of these components is illustrated by showing their performance for the 198 TREC-8 questions shown in Table 4.3. The results shown are for the long answer (250 bytes) task. The documents used are the top 30 retrieved by the ATT system, which was made available to the participants. Since 28 of the queries have no answer in the top 20, the best possible score is .859.

1)	Term matching	0.439
2)	Stemming	0.470
3)	Synonym	0.478
4)	RSV	0.498
5)	ICTF	0.509
6)	Exact	0.506
7)	Prox	0.515
8)	Head	0.515
9)	8)+Name heuristics	0.566
10)	8)+Numerical heuristics	0.584
11)	8)+Name+Numerical	0.616
12)	8)+Others	0.500
13)	8)+Others+Name+Num	0.589

Table 4.3 QA System for TREC-8 198 Queries

Until Line 8, there were steady improvements in the score when we augment the system with a new component. Line 12 shows, that when Others (Phrases Caps and Quoted described in number 9 10 and 11) are added in to the previous 8, overall performance is actually harmed. Unfortunately this was discovered too late and they were included in the official run.

4.3 Results and Discussions

Four runs named pir0qa[sl][12] were submitted. The s or l indicates short (50byte) or long (250 byte) answers. The submitted runs ending with 1 utilized the top 50 retrievals of the ATT system; the runs ending with 2 used the top 300 sub-documents retrieved by our PIRCS system. PIRCS preprocesses the original documents and returns sub-documents of about 300-550 words in size. Tag information such as heading and date are lost, which may result in small degradation of the final score. Table 4.4 compares the submitted runs to the TREC overall median result using 'strict' MRR evaluation. It seems to indicate that using more documents in the retrieval list

TREC long average	0.350	base
pir0qal1	0.433	+23.77%
pir0qal2	0.464	+32.73%
TREC short average	0.218	base
pir0qas1	0.263	+20.82%
pir0qas2	0.284	+30.65%

Table 4.4. MRR Comparison with TREC-9 Median

helps a lot (pir0q?2 vs pir0q?1). Our simple strategy returns results 20 –33% better than median.

We attempted to analyze our results to see what are the difficulties in QA in general.

Easy questions we missed
The queries may be ranked by their overall performance from all the participants. It is instructive to look at some easy queries that we missed. We comment mainly on pir0qal1, which uses the ATT retrieval list.

207. What is Francis Scott Key best known for?
This is a failure to recognize meta-words, words that are instructions to the query engine rather than real content words. We gave too much credit for matching best and known.

265. What's the farthest planet from the sun?
Our system returned Neptune, which at that time was the farthest. The high-scoring sentence 'Pluto, the farthest planet from the sun' from AP901116-0022 was not returned by the ATT retrieval within 30 documents. PIRCS returned this sentence, and pir0qal2 got full credit.

447. What is anise?
In this query, the name Anisi was confused with anise. Since this is a one-word query, the ranking was decided by the document RSV. Perhaps more credit should be given to exact match than stemmed match, or don't stem proper names at all.

500. What city in Florida is Sea World in?
We had Orlando in our answer, but it was judged incorrect.

504. Who is the founder of the Wal-Mart stores?
Our system did retrieve the correct sentence, but it was long and the correct phrase was not returned. Strangely pir0qas1, the 50-byte answer found the correct phrase.

683. What do river otters eat?
Oops, we again retrieved a correct sentence and filtered out the correct phrase.

688. What country are Godiva chocolates from?
Our system tries to match the word 'country'.

715. What could I see in Reims?
This is a difficult question.

Difficult questions
There are a number of queries for which NLP is required. Consider the following:

679. What did Delilah do to Samson's hair?
The answer to this can be found in the following three sentences: "Samson, whose story is told in the Book of Judges, was known for feats of enormous strength, such as slaying 1,000 Philistines with the jawbone of a mule. But he was stopped by Delilah, who was sent by the Philistines. She seduced him, learned that the secret to his strength was his hair and cut it off while he was sleeping." Impressively some systems were able to resolve the references and find the correct answer.

Some queries like:
208. What state has the most Indians?
375. What ocean did the Titanic sink in?
581. What flower did Vincent Van Gogh paint?
688. What country are Godiva chocolates from?

seek a specific class of objects. A good NLP system would make use of knowledge bases, listing states, countries, flowers and oceans. A naive retrieval system like ours, only matches the words state, flower, country and ocean.

Another difficult query is
471. What year did Hitler die?

The answer is in strings like 'the Nazi leader committed suicide April 30, 1945' and 'Hitler killed himself in 1945', which requires the knowledge that suicide and killed are a form of death.

The two senses of who
The word 'who' in a query has two meanings. Consider the queries:
209. Who invented the paper clip?
269. Who was Picasso?

The first question seeks a person, while the second is looking for a description. Our system assumes the first case. Table 4.5 shows that while this does not harm the long answer, it is disastrous for the short. Apparently, other participants had fewer problems with this. At any rate, this illustrates the dangers of applying highly specific heuristics.

	Num of queries	TREC long	TREC short	pirc0qal1	pirc0qas1
who/1	90	0.42	0.30	0.52	0.44
who/2	20	0.51	0.22	0.60	0.08

Table 4.5. Two Types of "Who"

5 Adaptive Filtering Track

This year, by some coincidence, all experiments we participated involve either new programs or heavy extensions to old programs. Moreover, we also took part in other cross language experiments that have deadline quite close to the filtering track. We found ourselves overextended both in time and resources. Some formatting of the OHSU collection for our system was done earlier, but at the end we found no time to do any training or testing. Finally, we decided to use our old programs from TREC-7 & 8 as is without change, and just release them on the OHSU data – to see how bad it gets without training at all. The parameters of the program were trained on newspaper type of documents, while the OHSU data is of course medical documents. One thing we did try to tailor to the new environment was to use the topic descriptions to do retrieval on OHSU87 documents, and expand the queries in a pseudo-relevance feedback fashion, but with the two given relevant documents included. Our filtering runs were supposed to target for utility values rather than precision. The resulting mean T9U score of –55.7and –69.14 were bad. Apparently, expanding the query at the beginning and running a system without training is not a good idea.

6 Conclusion

Our query translation approach to cross language retrieval by combining MT software and bilingual wordlist lookup with disambiguation seems to work quite well – at over 80% of monolingual effectiveness. This is because the topics do not carry too many names or proper nouns that are not translatable by our resources. There were only 25 queries for this experiment. More query types as well as document genre need to be experimented with in the future.

We have succeeded in extending our PIRCS system to handle 10 GB web data. This is done by aggressively screening away a lot of non-textual data. Results were well above median. For topics of a few words, it is necessary to devise ways to handle null queries – either due to spelling errors, or due to terms being filtered out due to high document frequencies.

We presented a QA system based on classical IR methods for sentence retrieval, enhanced with simple heuristics. It achieved above average results that can serve as a baseline. There is much room for future improvement. More heuristics, increased use of knowledge bases, exploring part-of-speech information and more careful query analysis may be employed to attain better performance.

Acknowledgment

This work was partially supported by the Space and Naval Warfare Systems Center San Diego, under grant No. N66001-1-8912. Peter Deng implemented a BIG5/GB conversion program for our use.

References

[BaCr98] Ballesteros, L and Croft, W.B. "Resolving ambiguities for cross-language retrieval." In: Proc. 21th Ann. Intl. ACM SIGIR Conf. on R&D in IR. pp. 64-71, 1998.

[LeSJ96] "Natural language processing for information retrieval", Comm. of ACM 39, pp.92-101, 1996.

[Kwok00] Kwok, K.L. "Exploiting a Chinese-English bilingual wordlist for English-Chinese cross language information retrieval' Proc. 5th Intl. Workshop on Information Retrieval with Asian Languages (IRAL'00), pp.173-179, 2000.

[KwCh98] Kwok, K.L & Chan, M "Improving two-stage ad-hoc retrieval for short queries." Proc. 21st Ann. Intl. ACM SIGIR Conf. on R&D in IR. pp.250-256, 1998.

[Pirk98] Pirkola, A. "The effects of query structure and dictionary setups in dictionary-based cross-language information retrieval." In: Proc. of 21th Ann. Intl. ACM SIGIR Conf. on R&D in IR. pp. 55-63, 1998.

Structuring and expanding queries in the probabilistic model

OGAWA Yasushi, MANO Hiroko, NARITA Masumi, HONMA Sakiko
Software Research Center, RICOH Co., Ltd.
1-1-17 Koishikawa, Bunkyo-ku, Tokyo 112-0002, JAPAN
{yogawa,mano,narita,honma}@src.ricoh.co.jp

1 Introduction

This is our second participation in TREC, following the last year's ad-hoc, and five runs were submitted for the main web track.

Our system is based on our Japanese text retrieval system [3], to which English tokenizer/stemmer has been added to process English text. Our indexing system stores term positions, thus providing proximity-based search, in which the user can specify the distance between query terms.

What our system does is outlined as follows:

1. Query construction

   The query constructor accepts each topic, extracts words in each of the appropriate fields and constructs a query to be supplied to the ranking system.

2. Initial retrieval

   The constucted query is fed into the ranking system, which then assigns term weights to query terms, scores each document and turns up a set of top-ranked documents assumed to be relevant to the topic (pseudo-relevant documents).

3. Query expansion

   The query expander collects and ranks the words in the pseudo-relevant documents and the words ranked the highest are added to the original query, with the words already in the query re-assigned new term weights.

4. Final retrieval

   The ranking system performs final retrieval using the modified query.

The basic features of the system are mostly the same as those implemented last year for the TREC-8 ad-hoc track [2]. In what follows, we explain each of the steps in more detail, both the features retained from last year and new enhancements we added this year for the TREC-9 main web track.

BEST COPY AVAILABLE

2 Query construction

We have employed automatic query processing to construct queries using single-word and phrasal search terms. In what follows, we describe how single and phrasal search terms are created by linguistic methods and represented as a structured query.

2.1 Basic features

2.1.1 Single term selection

Natural language text in each topic is processed by our English tokenizer and stemmer to output stemmed word tokens.[1] From the stemmed tokens, the query constructor selects single-word search terms by eliminating stopwords. We have used two kinds of stopword lists, the Fox's [1] word list for the <title> field and its augmented word list we created for the <desc> field.

2.1.2 Phrasal term selection

Noun phrases consisting of two or more single words are extracted for use in search terms. Syntactic phrases are recognized in the natural language text by applying the syntactic chunker LT_CHUNK developed at the Edinburgh Language Technology Group. Each noun phrase is then tokenized and stemmed. Phrases consisting of three or more single words are decomposed into sets of word pairs. For example, the noun phrase "industrial waste disposal" is decomposed to derive three word pairs "industrial waste," "waste disposal," and "industrial disposal."

2.1.3 Query representation

Single and phrasal search terms are combined into a query using syntax of our query language. Phrasal terms are represented using a proximity operator #WINDOW. For example, the phrasal search term "waste disposal" is represented as:

```
#WINDOW[1,1,o](waste,disposal)
```

where #WINDOW[1,1,o] specifies that the two component words are to be found adjacent and in the described order in a document. To deal with possible changes in word order and the number of intervening words between the component words of a phrasal term, we prepared a variant of the basic query representation, #WINDOW[2,*num*,u](A,B). The variant allows the words A and B to co-occur not in adjacency but within a specified number of words (*num*) of each other, in any order. We have also introduced a scaling operator #SCALE to phrasal term representation to adjust its term weight.

To sum, our sample queries are expressed as follows:

```
A query from the <title> and <desc> fields:
    #OR(killer,bee,attack,human,africanise,
        #SCALE[0.4](#WINDOW[1,1,o](killer,bee)),
        #SCALE[0.25](#WINDOW[2,50,u](killer,bee)))
```

---

[1] Text was used as is, with no spelling-correction applied.

BEST COPY AVAILABLE

2.2 Multiple fields

In TREC-8, the fields from which a word or phrase was extracted were not taken into account when a query is constructed. In TREC-9, when words are repeated across multiple fields in the topic, we increase their term weights to reflect their relative importance. To adjust the weights of these repeated words, the scaling operator #SCALE is applied and optimized to maximize the retrieval effectiveness.

```
Words extracted from the topic:
<title>: lava, lamp
<desc>: origin, operation, lava, lamp

Query:
#OR(origin,operation,#SCALE[3.0](lava),#SCALE[3.0](lamp))
```

3 Initial retrieval

For each query constructed, the ranking system ranks the documents in the target document collection and retrieves top-ranked documents. To rank documents, the system uses term weighting and document scoring formulae similar to Okapi's [4] but with some modifications.

The weight of each term is calculated by using the formula

$$w_t = \log\left(k_4' \cdot \frac{N}{n} + 1\right),$$

where N is the number of documents in the collection, n is the number of the documents in which the term occurs and k_4' is a parameter, with $0 \leq k_4'$.

Note that unlike Okaipi's, with our formula, the term weights never get negative. By keeping the term weights positive, the quality of retrieval is maintained even in the worst case.

With each term weighted according to the above formula, the ranking score for each document is calculated using the formula

$$s_{d,q} = \sum_{t \in q} \frac{w_t}{\log(k_4' \cdot N + 1)} \cdot \frac{f_{t,d}}{k_1((1-b) + b\frac{l_d}{l_{ave}}) + f_{t,d}}$$

where $f_{t,d}$ is the within-document frequency of the term, l_d is the document length, l_{ave} is the average document length, k_1 and b are parameters.

4 Query expansion

4.1 Basic features

The query expander, regarding the top-ranked documents as relevant documents, collects all single terms except stopwords in them and ranks the terms according to its Term Selection Value (TSV) while reweighting query terms, using formulae similar to Okapi's.

BEST COPY AVAILABLE

For each term collected, a new weight based on the feedback from the retrieved documents is assigned. The term reweighting formula reflects term weighting during initial retrieval mentioned above.

$$\begin{aligned} w_t \;=\;& \frac{k_5}{k_5+\sqrt{R}} \log\left(k'_4 \frac{N}{N-n} + \frac{n}{N-n}\right) + \frac{\sqrt{R}}{k_5+\sqrt{R}} \log \frac{r+0.5}{R-r+0.5} \\ & - \frac{k_6}{k_6+\sqrt{S}} \log \frac{n}{N-n} - \frac{\sqrt{S}}{k_6+\sqrt{S}} \log \frac{s+0.5}{S-s+0.5}, \end{aligned}$$

where R is the number of relevant documents, r is the number of relevant documents containing the term, S is the number of non-relevant documents, s is the number of non-relevant documents containing the term, and k_5 and k_6 are parameters. (S was set to 0 in the experiment.)

The query expander then calculates TSV for each reweighted term to select the terms to augument the query by using the formula

$$TSV = \left(\sum_{d \in R} \frac{f_{t,d}}{k_1((1-b) + b\frac{l_d}{l_{ave}}) + f_{t,d}} \Big/ R - \beta \cdot \sum_{d \in S} \frac{f_{t,d}}{k_1((1-b) + b\frac{l_d}{l_{ave}}) + f_{t,d}} \Big/ S \right) \cdot w_t$$

where β is a parameter.

Note the TSV formula has been changed from Okapi's to incorporate the within-document frequency.

The top-ranked single terms are then added to the original query with their respective term weights. The single terms and phrasal terms origianlly included in the query are also given re-assigned term weights, multiplied with a bonus factor.

4.2 Duplicates-resistant term selection

Compared with the ad-hoc track document collection we used last year, the web track document collection used this year seemed to contain far more documents that are exact or partial copies of some other documents. Although their presence in the retrieved documents may not affect retrieval effectiveness as measured by the current evaluation measures [6], it can degrade performance when the retrieved documents are used for automatic query expansion since this could give terms that appear in duplicates or near-duplicates an unjustifiably higher document frequency, thus boosting the likelihood of being selected.

To alleviate ill effects of duplicates and near-duplicates in the documents retrieved in initial retrieval, two work-arounds are devised:

1. During initial retrieval, eliminate documents scored the same as previously retrieved documents. If two documents have the same score, it is highly likely that they are exact copies of each other.
2. When selecting terms for expansion, for terms with a low document frequency among the retrieved documents, terms that appear in the same set of documents as those from which previously selected terms were drawn are excluded. These terms

are unlikely to co-occur in the same set of documents, and if they do, that may indicate the set shares the same piece of text.

4.3 Phrasal term addition

This year, we extend the idea of using phrasal terms from only in query generation to both in query generation and expansion. That is, when expanding queries, not only single words but also pairs of contiguous words in top-ranked documents are collected, evaluated and selected for use as expansion terms. The goal was to find a way to select the right pairs of words, with the right balance of weights, while keeping the expansion process simple and quick with minimal overhead.

In the experiments, we tested the same weighting/selection technique as used for single words for its applicability to word pairs and found it promising when the following adjustments were made:

- Set first the minimum Term Selection Value, which is set several times as high as that for single words.
- Give more importance to the document frequency in top-ranked documents compared with that in the document collection, when assigning a weight.
- Adjust the weights for word pairs that contain the same single words as those selected and those supplied in the original query.

5 Final retrieval

The expanded-and-reweighted query is sent to the ranking system and documents are retrieved as final result. Document ranking is done just as in initial retrieval, except that the term weights are supplied in the query.

6 Results

We tuned up the formulae using the WT2g document collection and mainly queries generated from the TREC-8 topics. Parameter values were chosen for each of the four categories below and are listed in Table 1 – 3.

- Queries using only <title> and queries using <title> and <desc>
- Queries using no phrasal search terms and queries using phrasal search terms

Note that in Table 1 and Table 2, we chose different sets of parameter values for the same retrieval parameters. This is because, for retrieval in a run with no query expansion, we wanted parameter values that would maximize average precision, whereas for initial retrieval for a run with query expansion, we looked for parameter values that would maximize precision at ten retrieved documents.

Table 1: Parameters for runs without expansion

	title only		title + desc	
	no phrases	phrases	no phrases	phrases
k_1	0.75	0.5	1.0	0.75
b	0.25	0.25	0.25	0.25
k'_4	0.5	0.5	0.5	0.5
Scale for multi-field query terms	–	–	3.0	3.0
num in #WINDOW[2,*num*,u]	–	50	–	10
Scale for #WINDOW[1,1,o]	–	0.25	–	0.4
Scale for #WINDOW[2,*num*,u]	–	0.1	–	0.25

Table 2: Parameters for runs with expansion (initial retrieval)

	title only		title + desc	
	no phrases	phrases	no phrases	phrases
k_1	1.5	0.75	1.5	0.75
b	0.25	0.25	0.25	0.25
k'_4	0.5	0.1	0.5	0.5
Scale for multi-field query terms	–	–	2.75	3.0
num in #WINDOW[2,*num*,u]	–	50	–	10
Scale for #WINDOW[1,1,o]	–	0.25	–	0.2
Scale for #WINDOW[2,*num*,u]	–	0.1	–	0.1

Table 3: Parameters for runs with expansion (final retrieval)

	title only		title + desc	
	no phrases	phrases	no phrases	phrases
Maximum number of documents used for expansion	10	10	10	10
k_1	0.75	0.75	1.0	0.75
b	0.25	0.25	0.25	0.25
k'_4	0.01	0.01	0.01	0.01
k_5 for single expansion terms	6.0	6.0	6.0	6.0
k_5 for phrasal expansion terms	2.0	3.0	3.0	3.0
Scale for multi-field query terms	–	–	3.0	3.0
num in #WINDOW[2,*num*,u]	–	50	–	10
Scale for #WINDOW[1,1,o]	–	0.4	–	0.3
Scale for #WINDOW[2,*num*,u]	–	0.25	–	0.2
Maximum number of terms to be added	25	25	30	30
Minimum number of terms to be added	10	10	10	10
Minimum r for term to qualify	2	2	2	2
Maximum r for near-duplicate checking	3	3	3	3
Scale for phrasal expansion terms	0.3	0.2	0.3	0.3
Minimum TSV factor for phrasal expansion terms	5	3.3	2.5	2.5
Bonus factor for query terms	10.0	10.0	7.0	5.0

Table 4: Average precision for TREC-8 topics

	title only		title + desc	
	no phrases	phrases	no phrases	phrases
baseline	0.3184	0.3247	0.3017	0.3113
above + multi-field scaling	–	–	0.3321	0.3420
above + expansion	0.3493	0.3536	0.3637	0.3703
above + phrasal expansion terms	0.3538	0.3533	0.3671	0.3716

The experimental runs using the above parameters, the TREC-8 topics and the WT2g document collection resulted in the following average precision measurements (Table 4). The table shows improvement in performance as more features are added.

Using the same conditions as above, the results for the TREC-9 topics and the WT10g document colletion are shown in Table 5. The runs submitted are indicated by asterisks.[2] (The numbers in the parentheses below the table are from the original submissions, which contained incorrect results due to program bugs. The numbers in the table were obtained after bug fixes and a minor modification in which the minimum number of words to be added was lowered to 0.)

---

[2] ric9dpxL is a linear combination of ric9dpx and HITS [5]. This run is yet to be analyzed.

BEST COPY AVAILABLE

Table 5: Average precision for TREC-9 topics

	title only		title + desc	
	no phrases	phrases	no phrases	phrases
baseline	0.2025	0.2073	0.2135	0.2407
above + multi-field scaling	–	–	0.2489	0.2608^{*3}
above + expansion	0.1740	0.2021	0.2212^{*2}	0.2427
above + phrasal expansion terms	0.1788	0.2034^{*1}	0.2211	0.2411^{*4}

^{*1}: ric9tpx (0.1787), ^{*2} ric9dsx (0.2201), ^{*3}: ric9dpn (0.2616), ^{*4}: ric9dpx (0.2267)

The results show that use of phrasal search terms and multi-field scaling worked well in TREC-9, as in TREC-8 above. However, unlike in TREC-8, we see that query expansion, whether with phrasal expansion terms or not, made unexpectedly negative effect in TREC-9. Why the difference?

One thing we noticed is the difference in the size of the target collection from which documents were retrieved. The WT10g document collection used for the TREC-9 submission had more than six times the number of documents in the WT2g document collection used for the TREC-8 experiments. The largeness of the WT10g collection led to rare words getting term weights that were extremely high because of the way we calculated the term weights, thus making it easier for these words to be selected as expansion terms.

Another difference is in the number of relevant documents for each of the topics. Although on the average, the number of relevant documents in TREC-9 is greater than that in TREC-8, there are more topics having a very few relevant documents in TREC-9 than in TREC-8; topics having fewer than 10 relevant documents add up to 11 in TREC-9, as opposed to only 2 in TREC-8. The system, on the other hand, retrieved just as many documents for these topics despite this, turning up fewer relevant documents in the top-ranked documents in initial retrieval before expansion. For example, P@10, or precision after 10 documents are retrieved, is 0.3520 in TREC-9, compared to 0.5000 in TREC-8, in runs in "title + desc, phrases." What this means is that the query expander had to select expansion terms from mostly non-relevant documents, which in addition may contain a near-duplicate or two skewing the term selection statistics even further in the wrong direction.

In the follow-up experiments conducted after the submission, we tested an alternative approach where expansion terms were selected without regard to their term weights so that their influence on term selection would be eliminated. The average precision we obtained from this approach in a run in "title + desc, phrases," was 0.2629, showing an 8% improvement from the submitted run.

References

[1] C. Fox. A stop list for general text. *ACM SIGIR Forum*, Vol. 24, No. 2, pp. 19–35, 1991.

BEST COPY AVAILABLE

[2] Y. Ogawa, H. Mano, M. Narita, and S. Honma. Structuring and expanding queries in the probabilistic model. In *The Eighth Text REtrieval Conference (TREC-8)*, 2000.

[3] Y. Ogawa and T. Matsuda. An efficient document retrieval method using n-gram indexing (in Japanese). *Transactions of IEICE*, Vol. J82-D-I, No. 1, pp. 121–129, 1999.

[4] S.E. Robertson and S. Walker. On relevance weights with little relevance information. In *Proc. of 20th ACM SIGIR Conf.*, pp. 16–24, 1997.

[5] M. Toyoda and M. Kitsuregawa. Finding related communities on the Web. In *Poster in WWW-9 Conf.*, 1999.

[6] D. Hawking, E. Voorhees, N. Craswell, and P. Bailey. Overview of the TREC-8 Web Track. In *The Eighth Text REtrieval Conference (TREC-8)*, 2000.

Support for Question-Answering in Interactive Information Retrieval: Rutgers' TREC-9 Interactive Track Experience.

N.J. Belkin, A. Keller, D. Kelly, J. Perez-Carballo, C. Sikora*, Y. Sun
School of Communication, Information & Library Studies
Rutgers University
4 Huntington Street
New Brunswick NJ 08901-1071
nick@belkin.rutgers.edu [amkeller | dianekel | carballo | ysun]@scils.rutgers.edu
csikora@lucent.com

Abstract

We compared two different interfaces to the InQuery IR system with respect to their support for the TREC-9 Interactive Track Question-Answering task. One interface presented search results as a ranked list of document titles (displayed ten at one time), with the text of one document (the first, or any selected one) displayed in a scrollable window. The other presented search results as a ranked series of scrollable windows of the texts of the retrieved documents, displayed six documents at a time, each document display beginning at the system-computed "best passage". Our hypotheses were that: multiple-text, best passage display would have an overall advantage for question answering; single-text, multiple title display would have an advantage for the list-oriented question types; and that multiple-text, best passage display would have an advantage for the comparison-oriented question types. The two interfaces were compared on effectiveness, usability and preference measures for sixteen subjects. Results were equivocal.

1 Introduction

The TREC-9 Interactive Track (IT) changed the searching task from the instance/aspectual recall task used in the previous three TRECs, to a question-answering task. This new task, although drawing upon the same database as that of the Question-Answering (Q-A) Track, differed substantially from the Q-A Track task, in that the questions that the subjects were to answer were designed to require more than one document in order to be correctly answered. Furthermore, questions were constructed as two types: one which asked for a list of items as an answer (e.g. what are three national parks in which one can find redwood trees?), the other which required comparison of items for an answer (e.g. is Denmark larger than Norway in population?).

At Rutgers, we decided to investigate the support of people trying to answer questions of these two types through interface design. We supposed that an interface which allowed viewing of more than one document text at time would be beneficial for comparison-type questions, since that might make it easier for the searcher to make the necessary comparisons. We further supposed, based on our experience in supporting the instance recall task in previous IT experiments, that an interface which showed many possibly useful documents at once would be beneficial for the listing-type question, and that this could be accomplished through reasonably informative document surrogates, rather than the texts. Finally, we supposed that, in order to support question-answering in general, helping the person to get to the most relevant part of a document (i.e., where some part of the answer was likely to be located) would be beneficial. In part, this idea is based on the approaches and results of the Q-A Track in previous TRECs, since performance was quite high for most systems when 250 bytes were retrieved.

We translated these suppositions into interface designs, and related hypotheses, which could be investigated within the structure of the IT. For the first supposition, we used an interface which we had developed for the TREC-8 instance recall task, since that task shares a number of features in common with the listing-type question. We named this the SDD system (see section 3, below, for details of both systems implemented in this study, and Belkin, et al., 2000 for a description of our TREC-8 study). For the second supposition, we implemented an interface with the same functionality as the SDD system, but which displayed, in a scrollable "document display window", six scrollable panes containing the texts of the six retrieved documents from the selected part of the retrieved document list. And in response to our third supposition, the document texts in this system, which we named MDD, were displayed beginning at the system-determined best passage, rather than at the beginning of the document, as in SDD.

The hypotheses that we tested in this study were, thus:

Hypothesis 1: MDD will support the comparison-type task better than SDD, where "better" is measured in terms of performance and effort.

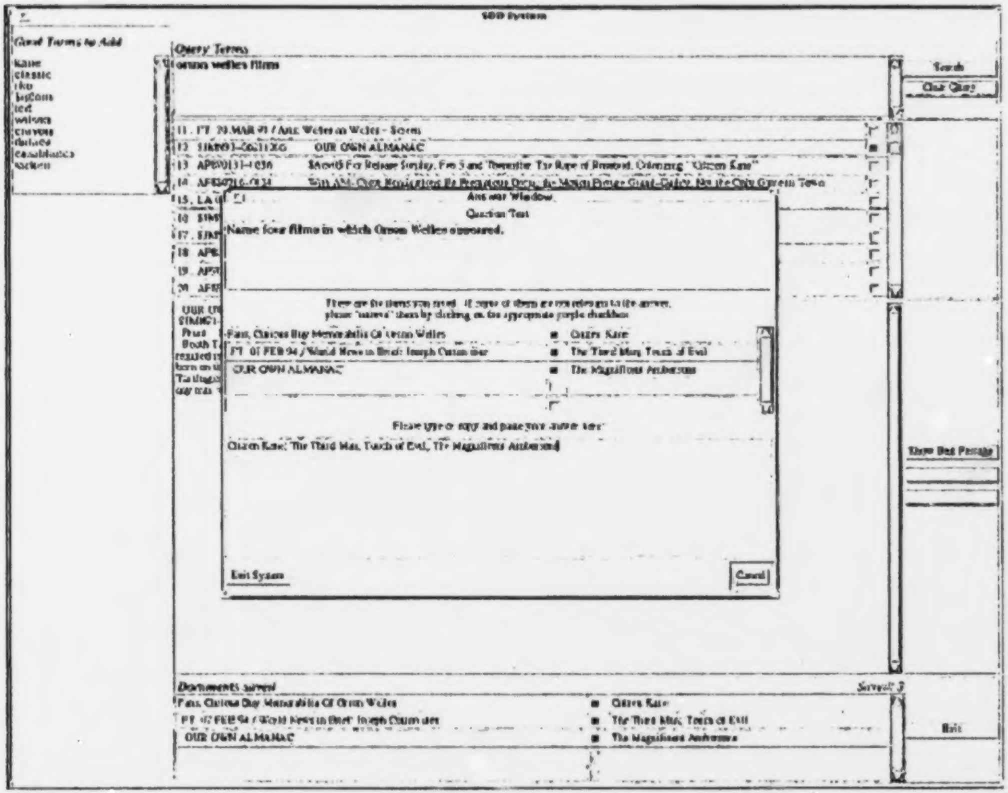
Hypothesis 2: SDD will support the listing-type task better than MDD (measured as in Hypothesis 1)

Hypothesis 3: MDD will support the question-answering task overall (i.e. both tasks combined) better than SDD, where "better" is measured in terms of performance, effort, and user preference.

2 System descriptions

There were two experimental IR systems used in this study. Both systems used Inquery 3.1p1 with its default values for indexing and retrieval (cf. Callan, Croft & Harding, 1992). A SUN Ultra-1 with 512MB memory and 9GB disk under Solaris 2.5.1 with a 20" color monitor was used with both systems. The primary difference between the two systems involves the layout of the information associated with the documents retrieved. This difference results in disparities in the type and amount of information displayed, and associated interactions with that information.

The first system, Single Document Display (SDD), presented the top ten document titles and the text of the first document. The text window displayed 32 lines of text and extended most of the width of the screen. The document text was positioned at the beginning of the document. Users could move quickly to the best passages in the text by using the **"Show Best Passage," "Show Next Best"** and **"Show Prev Best"** buttons located next to the document text window. "Good" passages and their ranks with respect to one another were determined according to the InQuery 3.1p2 default values, with the length of passage set to 20 words. Clicking on a different title in the list provided the text of that document in the document window. Scrolling the title list provided new document titles. A document could be saved or unsaved by clicking on a toggle checkbox located to the right of each document title. The SDD interface is shown in Figure 1.



Name four films in which Orson Welles appeared.

Figure 1: SDD system with final "**Answer Window**" displayed.

The second system, Multiple Document Display (MDD), presented the title and text of the top six documents in a format consistent with that used by Golovchinsky and Chignell (1997). Two rows of three document windows were displayed across the entire width of the screen. Each document window displayed 21 lines of text under a title bar that displayed information about the document ID and a truncated document title. The document text is positioned such that the best passage is displayed at the top of the text window. Users could move to other good passages within the document by using the "**Next Pass**" and "**Prev Pass**" buttons located below each document window. Next to those buttons, there is also a button labeled "**Top**" to allow the user to jump to the beginning of the document text. Each text window had a scrollbar to move up and down throughout the text. A scrollbar at the side of the screen allows the user to view other documents. There is a button at the bottom of each text window to "**Save**" that document. The button changes to "**Unsave**" to allow users to change the status of the document. Figure 2 is a screenshot of the MDD interface.

BEST COPY AVAILABLE

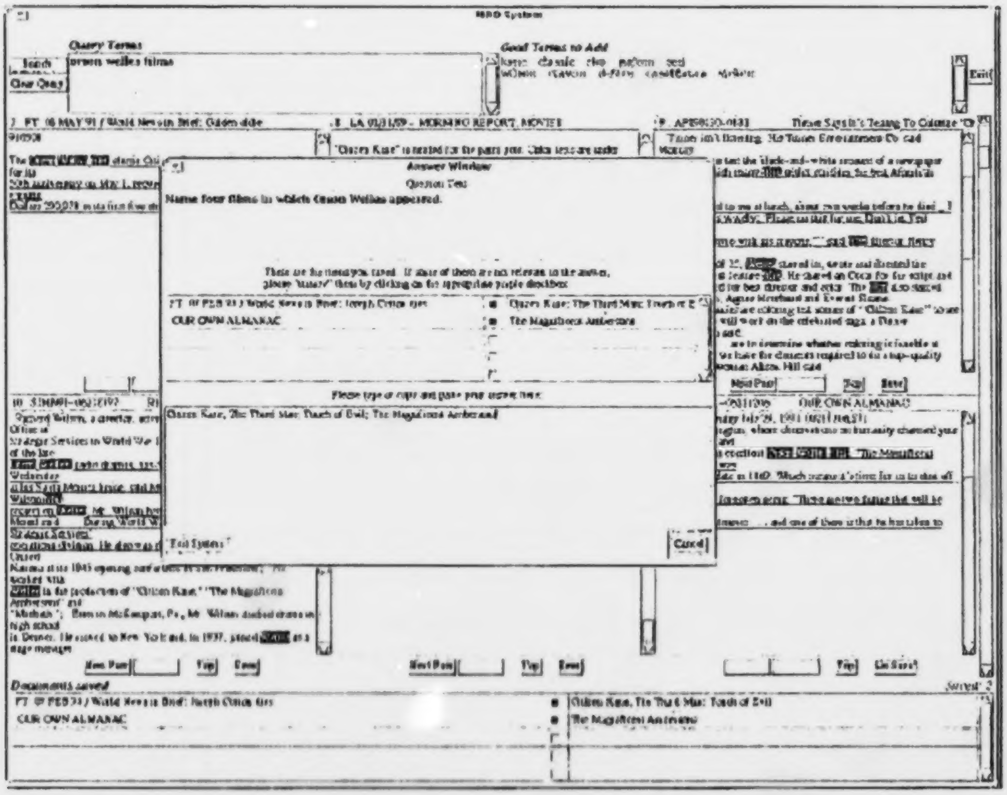


*Figure 2: MDD system with final "**Answer Window**" displayed.*

The interface features of both systems were similar and are described below:

- **Query Terms** Window – A window at the top of the application that was used to input a free-form query. It did allow for minimal structure (e.g., phrases).
- **Good Terms to Add** Window – A display window next to the "**Query Terms**" window provided suggested good terms to add to the query. The user could click on a term to add to the query window for the next search iteration. These terms were determined using pseudo-relevance feedback, based on the first ten documents displayed, and using the default relevance feedback formula for InQuery 3.1p2. The top ten relevance feedback terms were then entered into this window.
- **Pop-Up Answer Window** – A dialog box that appeared when a document was saved that required the user to label the saved document with the portion of the answer that it represented.
- **Documents Saved** Window – A display window at the bottom of the screen that provided a list of the document titles of the saved documents. Clicking on the title displayed the document text. The user could unsave the document by clicking on the check box located to the right of each saved document title.

BEST COPY AVAILABLE

- **Document Label** Window – A display window to the right of the "**Documents Saved**" window that displays the label associated with each saved document. To edit the label the user clicks on the label.
- **Search** Button – A button used to initiate the search based on the terms in the "**Query Terms**" window, which generated the documents retrieved.
- **Clear Query** Button – A button used to remove all of the terms in the "**Query Terms**" window.
- **Exit** Button – A button used to end the search session.
- Final **Answer Window** – A dialog box was presented at the end of the search to allow users to type in their final answer. The window also presented the search question, the saved documents and the associated labels for those documents. The user was allowed to click on the titles to see the text.
- **Stop Search** Window – A window that covered the entire screen at the end of five minutes alerting the user that the time was up.

3 Methods

We followed strictly the TREC-9 Interactive Track protocol for this experiment (see Over & Hersh, this volume, for a complete description of the experimental design). This protocol required a minimum of 16 subjects, each of whom searched the same database in order to answer four questions using one system, and then another four questions using the other system. Questions (also called topics) were divided into two categories: listing-type questions (topics numbered 1-4), and comparison-type questions (topics numbered 5-8).

A total of 16 volunteer subjects, recruited informally by the experimenters, participated in this project. A majority (81%) of the subjects either held, or were expecting, graduate-level degrees from varying disciplines such as law, library studies, and women's studies. The remaining participants had obtained a bachelor's degree and were employed in fields from librarianship to civil engineering. None had taken part in previous TREC studies. Each subject conducted eight searches in accordance with the TREC-9 Interactive Track experimental guidelines. Subjects conducted four searches in both the MDD and SDD systems. We used a Latin square design where eight topics were randomized and rotated completely so that each topic appeared only once in each row and once in each column. The same set of topics was rotated again with a different system order, in order to allow a direct comparison between two different systems. Sixteen different combinations of topic order and system order were used allowing us to run experiments with 16 subjects.

On arrival, the subjects read and signed a consent form explaining their rights and potential risks associated with participation in the experiment. They then completed a demographic questionnaire that gathered background information and probed their previous searching experience. Next, they received a hands-on tutorial for the first system, describing the various features of that system. After completing the tutorial, subjects were given a general task description and were told that they would have five minutes in which to execute each search, and that they would be warned by the experimenter when only one minute of search time remained. Before each question, participants were asked to provide an answer to the question, if they thought they knew it, and to indicate their degree of confidence in the answer. After five minutes, the system prompted the subjects to answer the question. As they searched, participants labeled aspects of answers to the questions as they identified them and saved documents. During the search sessions, they were asked to continuously "think aloud." A videotape recorded the computer monitor during both the tutorial and search portions of the experiment in order to capture all "thinking aloud" utterances. The entire session, of tutorial and searches, was logged.

After conducting each search, subjects answered several questions about their familiarity with the search topic, experiences with the searching task, their satisfaction with the search result, and satisfaction with the amount of time allotted for the search. After completing four searches for the first system, subjects answered several questions about the system in general. After a short break, the subjects were given a tutorial for the second system, searched another four topics, a pre-search evaluation and post-search questionnaire for each topic, and a post-system questionnaire. After completing all eight searches, the subjects completed an exit interview. The entire session took between 2 and 2 1/2 hours.

As mentioned above, most (81%) of the subjects either currently held or expected to receive graduate degrees, and the rest held bachelors degrees and were employed in various areas of the work force. Slightly more than half (56%) of the subjects were male. The average age of the subjects was 37. Half (50%) stated their primary occupation as student. On average, these searchers had been doing online searching for just over five years (M=5.56).

We asked a series of questions about the background experiences of our searchers, using a 5 point scale, wherein 1=no experience and 5=a great deal of experience. Overall, the searchers were quite familiar with the use of GUIs (M=4.88) and with Web search engines (M=4.56). A majority reported having had some experience with OPACs (M=4.19) and with searching on CD ROM systems (M=3.3).

Of note is that experience searching on commercial online systems in general was reported to be fairly low for our subjects (M=2.6), and experience searching on systems other than the Web was markedly low (M=1.6). On a final note, the searchers in our study tended to say that they enjoyed conducting information searches (M=4.2) as measured by a 5 point scale wherein 1=strongly disagree and 5=strongly agree.

4 Results

4.1 General

The two systems were compared according the three criteria: performance, effort (a measure of usability), and preference. Performance was measured on a binary scale: if the question was both completely answered, and correctly supported, then the answer was correct; otherwise, the answer was incorrect. Effort was measured in a variety of ways, including search time, number of cycles per search, and various measures indicating amount of interaction. Preference was measured by questions eliciting subject evaluation of the two systems. The results of the experiment are presented in the following sections, arranged according to each of our three hypotheses.

The overall data on correct answers, by subject, topic and system, are shown in Table 1. Seven of our subjects answered four of the eight questions correctly; two answered three correctly; four had two correct answers; two had one correct answer, and one had no correct answers. Three topics, numbers 1, 3 and 6, had no correct answers, and of these, two were of the list-type. These three could be termed "hard" questions for our searchers. Topic 5 had thirteen correct answers and topic 7 fourteen: these were "easy" questions for our searchers. Overall, each system provided the same number of correct answers. Topics 2, 4 and 8 were, by this system, "moderately difficult".

SUBJECT NO.	TOPIC NO.								
	1	2	3	4	5	6	7	8	TOTAL
1		SDD			MDD		MDD	MDD	S=1 M=3
2				MDD	SDD		SDD	MDD	S=2 M=2
3				SDD	MDD		MDD	MDD	S=1 M=3
4							MDD		M=1
5		MDD			SDD				S=1 M=1
6					SDD		SDD		S=2
7				MDD	MDD		SDD		S=1 M=2
8							SDD		S=1
9									0
10		SDD		SDD	MDD		MDD		S=2 M=2
11		SDD		MDD	SDD		SDD		S=3 M=1
12					MDD		SDD		S=1 M=1
13		SDD		SDD	MDD		SDD		S=3 M=1
14		SDD		SDD	MDD		MDD		S=2 M=2
15					SDD		MDD	MDD	S=1 M=2
16					SDD		MDD		S=1 M=1
TOTALS	0	S=5 M=1	0	S=4 M=3	S=6 M=7	0	S=7 M=7	S=0 M=4	S=22 M=22

Table 1. Correct answers by each subject, for each topic, indicating system used.

4.2 *Hypothesis 1: MDD supports the Comparison-type task better than SDD*

4.2.1 Performance

Performance on the comparison-type task was measured by number of correct, fully supported responses to topics 5-8. The means and standard deviations for these performance measures are displayed in Table 2. For all 16 subjects, the mean number of correct fully supported responses for the two systems was close (MDD: M = 1.13, SD=1.02; SDD: M=.88, SD=.72). The difference was not significant [t(15) = .66, ns]. For the 7 high performers (defined as those subjects who got at least a total of 4 correct, fully supported responses), the mean number correct on topics 5-8 for MDD system was nearly twice the number for SDD system (MDD: M = 1.71, SD= 1.11; SDD: M = .86, SD=.90). However with a size of 7 cases, the difference was not significant [t(6) = 1.16, ns]. For the 9 low performers (subjects with at most 3 correct fully supported responses of the total 8 questions), the means were similar and the difference was not significant [MDD: M = .67, SD=.71; SDD: M = .89, SD=.60; t(8) = -.69, ns]. All the non-significant differences suggest that the effectiveness of the two systems for the comparison-type task is similar. There was no system order effect for these results.

	TOTAL M (SD)	**MDD** M(SD)	**SDD** M(SD)
All subjects (N=16)	1.01(.87)	1.13(1.02)	.88(.72)
High performers (N=7)	1.29(1.01)	1.71(1.11)	.86(.90)
Low performers (N = 9)	.78(.66)	.67(.71)	.89(.60)

Table 2. Means and Standard Deviations of Comparison-type Task Performance

4.2.2 Effort

Effort was measured by searching time, number of cycles, and effort associated with interacting with the two systems for each comparison-type question.

For the comparison-type task (topics 5-8), the number of cycles in a search was roughly the same for the two systems (MDD: M = 2.16, SD=1.08; SDD: M = 2.66, SD=1.62). The difference is not significant [t(62)=.15, ns].

The average time used in a single search for the two systems was close (MDD: M=300.97 seconds, SD=123.84; SDD: M=326.97 seconds, SD=93.43). The difference was not significant [t(62)=-.95, ns].

The effort associated with interacting with the two systems was different as measured by scrolling behavior and use of the document navigation facilities (Next Passage, Best Passage and Top of Document). The difference between systems was significant for the number of times subjects used the scrolling feature in each of the systems [MDD: M=102.59, SD=175.22; SDD: M=32.63, SD=42.46; t(62)=2.20, p<.05]. The use of the Next Passage, Best Passage and Top of Document navigation facilities yielded significant difference between the two systems [MDD: M=3.28, SD=4.44; SDD: M=.25, SD=.51; t(62)=2.69, p<.01]. These results suggest that MDD required more effort than SDD, for similar performance.

Thus, based on performance and effort measures, we conclude that Hypothesis 1 is not supported.

4.3 Hypothesis 2: SDD supports the listing-type task better than MDD.

4.3.1 Performance

Performance on the listing-type tasks was measured by the number of correct, fully supported responses to topics 1-4. The means and standard deviations for these performance measures are displayed in Table 3. For all 16 subjects, the mean number of correct, fully supported responses for MDD was half of that for SDD (MDD: M= .25, SD= .45; SDD: M= .50, SD= .73). However the difference was not significant [t(15) = -1.07, ns]. For the 7 high performers defined previously, the mean performance for the SDD system was almost four times that of the MDD system (MDD: M = .29, SD= .49; SDD: M = 1.14, SD= .69). However with a small size of 7 cases, the difference was not significant [t(6) = -2.12, ns]. For the 9 low performers, no one had a correct, fully supported response with the SDD system (M = .00, SD= .00), and the mean number for MDD was .22 SD= .44). The difference was not significant [t(8) = 1.51, ns]. The non-significant results suggest that for the listing-type task the effectiveness of the two systems is similar. There was no system order effect on these results.

	TOTAL M (SD)	**MDD** M(SD)	**SDD** M(SD)
All subjects (N=16)	.38(.59)	.25(.45)	.50(.73)
High performers (N=7)	.72(.59)	.29(.49)	1.14(.69)
Low performers (N = 9)	.11(.22)	.22(.44)	.00(.00)

Table 3. Means and Standard Deviations of List Task Performance

4.3.2 Effort

For the listing-type tasks, the mean number of cycles in a search for SDD was more than MDD (MDD: M = 1.78, SD=1.01; SDD: M = 2.19, SD=1.64). This result is not significant [t(62)=-1.20, ns], therefore does not support our hypothesis.

The average time used in a single search for the two systems was roughly the same (MDD: M=367.16 seconds, SD=88.77; SDD: M=366.22 seconds, SD=130.15). The difference was not significant [t(62)=.03, ns]. Combined with the result of time measure presented in section 4.1.2, we can see that when searching, most subjects used the entire five minutes regardless of which system they were using and which type of question they were answering.

The effort associated with interacting with the two systems was different based on scrolling behavior and use of the document navigation facilities (i.e. Next Passage, Best Passage and Top of Document). The difference between the two systems was significant for the number of times subjects used the scrolling feature in each of the systems (MDD: M=91.13, SD=115.29; SDD: M=40.28, SD=44.52; t(62)=2.33, p<.05). The use of the Next Passage, Best Passage and Top of Document navigation facilities yielded significant differences between the two systems [MDD: M=3.59, SD=7.12; SDD: M=.19, SD=.90; t(62)=2.69, p<.01]. This suggests that there were more interactions with MDD than SDD in a single search, which is in accord with what we predicted. There were fewer interactions with SDD in both task types, suggesting that navigation use is consistent in both task types.

Based on performance, and to some extent on effort, we conclude that Hypothesis 2 is not supported.

4.4 *Hypothesis 3: Starting the document display at the best passage (MDD) is better than starting the document display at the top of the document (SDD).*

4.4.1 *Performance*

There was no significant difference between the number of correct, fully supported answers that were found using MDD and those found using SDD. Indeed, performance was nearly identical (MDD: M=1.38, SD=.96; SDD: M=1.38, SD=.89; t (15) = .00, ns). Additionally, there was no order effect for performance (MDD-SDD: M=2.75, SD=1.16; SDD-MDD: M=2.75, SD=1.58; t (15) = .00, ns).

4.4.2 *Effort*

Effort was measured by time, number of cycles, scrolling behavior and use of the document navigation facilities (i.e. Next Passage, Best Passage and Top of Document). The means and standard deviations for each of these measures are displayed in Table 4.

There was no significant difference between the amount of time users spent searching in each system. The means and standard deviations were roughly equivalent (MDD: M=334.06 seconds, SD=111.96; SDD: M=346.59, SD=114.11; t (126) = -.627, ns), suggesting that when searching, most subjects used the entire five minutes regardless of which system they were using.

Subjective measures of searching time and satisfaction with results indicated that, on a 5 point Likert scale, where 1=not at all, 3=somewhat and 5-extremely, subjects felt that they had somewhat enough time to conduct an effective search in each of the systems (MDD: M=3.13, SD=1.45; SDD: M=3.05, SD=1.41; t (125) = .305, ns) and that they were somewhat satisfied with their search results in each of the systems (MDD: M=3.23, SD=1.55; SDD: M=3.00, SD=1.55; t (125) = .853, ns).

While there was no significant difference for amount of time spent searching in each of the systems, there was a significant difference between the number of cycles. The mean number of cycles for MDD was 1.97 (SD=1.05) and the mean number for SDD was 2.42 (SD=1.63), t (126) = -1.87, p<.01. There was also a significant difference [t (126) = 2.55, p<.01] between the number of cycles for searches resulting in correct answers and incorrect answers. During searches that resulted in correct answers, subjects completed an average of 1.77 cycles (SD=1.05). During those searches that resulted in incorrect answers, subjects completed an average of 2.42 cycles (SD=1.49).

Interaction with documents was measured by scrolling behavior and use of the document navigation facilities (Next Passage, Best Passage and Top of Document). There was a significant difference in the number of times subjects used the scrolling feature in each of the systems (MDD: M=96.86, SD=147.24; SDD: M=36.45, SD=43.33; t (126) = 3.15, p<.00. There were also significant differences in the number of times subjects used the document navigation facilities for each of the systems. The mean use of document navigation facilities in MDD was 2.98 (SD=5.92), while their use in SDD was .22 (SD=.72), t (126) = 3.75, p<.00. However, these results should be interpreted with care as each system began the document display at a different position in the document. Each system also provided different opportunities for scrolling.

	MDD *M (SD)*	**SDD** *M (SD)*
Time (in seconds)	334.06 (111.96)	346.59 (114.11)
Cycles	1.97 (1.05)	2.42 (1.63)
Scrolling	96.86 (147.24)	36.45 (43.33)
Document Navigation Facilities	2.98 (5.92)	.22 (.72)

Table 4. Means and stand deviations associated with Effort Measures
Note: For MDD and SDD, n=64. Each mean based on one search topic session.

4.4.3 Preference

System preference with regard to where the document display began, either at the best passage (MDD) or at the top of the document (SDD) was measured by subjective responses to two questions presented at the end of the experimental session. These questions were "Which did you find most helpful for this task?" and "Which of these did you like best?" The responses to these questions indicated that there was no significant difference in preference for where the document display began, MDD (44%) and SDD (56%), χ^2= 3.88, ns and MDD (56%) and SDD (44%), χ^2=3.50, ns, respectively. Neither of these measures was significantly related to system order or to performance.

Subjects responded to several questions about where the document display began in the post-system questionnaires. During the MDD post-system questionnaire, subjects responded to the following questions, "How useful was it to have the documents start with the best passage?" and "How useful was the title information in finding an answer to the question?" During the SDD post-system questionnaire, subjects responded to the following questions, "How useful was the best passage feature?" and "How useful was the title list in finding answers to the question?" In all cases, subjects responded using a 5-point Likert scale, where 1=not at all, 3=somewhat and 5=extremely. Although there was no significant difference between subjects' responses to the two questions regarding the usefulness of the best passage feature [M=2.80, SD=1.32; M=2.87, SD=1.60, respectively, t (15) = -.14, ns], there was a significant difference between subjects' responses to the questions regarding the usefulness of the title information and title list. Specifically, subjects rated the title list in SDD (M=.31, SD=1.01), as significantly more useful in finding answers to questions than the title information in MDD (M=3.19, SD=.91), t (15) = -2.67, p<.05. There were no order effects for any of these responses.

During the post-system questionnaires, subjects also responded to questions regarding their perceptions of the usefulness of the document display method. In the MDD post-system questionnaire, subjects were asked, "How useful was it to see six documents at a time?" In the SDD post-system questionnaire, subjects were asked, "How useful was the document display in finding answers to questions?" In both cases, subjects responded using a 5-point Likert scale, where 1=not at all, 3=somewhat and 5=extremely. The results indicated that subjects rated the document display method of SDD (M=4.13, SD=.72) as significantly more useful than the document display method of MDD (M=3.5, SD=1.26), t (15) = -2.2, p<.05. Interestingly, this preference for one document display method over another was not observed in the exit interview questions reported in the preceding paragraph.

On the basis of these results, we cannot conclude that Hypothesis 3 is not supported, since there was some advantage to MDD in effort as measured by number of cycles. However, it would be inappropriate to believe it to be strongly supported.

5 Discussion and conclusions

In general, it seems that we are forced to conclude, based on the data analyses carried out so far, that two of our hypotheses are fairly conclusively rejected, while the third is at best partially supported. That is, there seems to have been no advantage in the comparison task to the MDD system, no advantage in the listing task to the SDD system, and only minor advantage to the MDD system overall.

However, one fairly significant analysis with respect to effort, comparing documents viewed and seen in the two systems remains to be done. In addition to any possible direct results from this analysis, it is possible that it may also temper the results having to do with scrolling behaviors in texts within texts. Also, we note that for both hypotheses 1 and 2 there were some large, but not significant differences between the two systems in the predicted directions. Unfortunately, the number of subjects was so small that these differences were not significant. Finally, because of resource constraints, our experiment was not quite as clean as we could have wished. The "multiple document display" supposition, and the "best passage" supposition were confounded in the experiment, since they were both implemented only in the MDD system. So we're not quite willing to give up yet.

Right now, we are still thinking about these results, and trying to understand why they seem to have run so counter to our initial intuitions. We hope to provide some answers to these questions at the meeting.

6 References

Belkin, N.J., Cool, C., Head, J. Jeng, J., Kelly, D., Lin, S.J., Lobash, L., Park, S.Y., Savage-Knepshield, P. & Sikora, C. (2000) Relevance feedback versus Local Context Analysis as term suggestion devices: Rutgers' TREC-8 Interactive Track experience. In *TREC-8. Proceedings of the Eighth Text Retrieval Conference*. Washington, D.C.: GPO, in press.

Callan, J.P., Croft, W.B. & Harding, S.M. (1992) The INQUERY retrieval system. In *Dexa 3*, Proceedings of the Third International Conference on Database and Expert System Applications. Berlin: Springer Verlag, 78-83.

Golovchinsky, G. and Chignell, M. H. (1997). The newspaper as an information exploration metaphor, *Information Processing & Management, 33 (5)*, 663-683.

Over, P. & Hersh, W. (this volume). Overview of the TREC-9 interactive track.

Logical Analysis of Data in the TREC-9 Filtering Track

Endre Boros[a], Paul B. Kantor[a,b] and David J. Neu[a]
[a]RUTCOR, Rutgers University
[b]SCILS, Rutgers University
{boros,neu}@rutcor.rutgers.edu, kantor@scils.rutgers.edu

Abstract

In the TREC-9 adaptive filtering and routing sub-tasks of the filtering track we continued to explore utilizing the Logical Analysis of Data (LAD) machine learning methodology to develop Boolean expressions that can be utilized as "rules" for characterizing relevant and irrelevant documents.

1 Logical Analysis of Data

In TREC-9, we continue to view the filtering and adaptive tracks as classification problems which can be approached via machine learning techniques. Specifically, we experiment with a machine learning methodology called Logical Analysis of Data (LAD) which was developed at the Rutgers Center for Operations Research (RUTCOR) at Rutgers University [6], [2], [1], [3], [4], [5]. We begin by providing a brief overview of LAD and then discuss how LAD was adapted for utilization at TREC-9.

LAD accepts as input a training set, each element of which is known to be either a *positive* instance or a *negative* instance. In the filtering and adaptive tracks, the positive instances correspond to *relevant* documents and the negative instances correspond to *irrelevant* documents. We shall refer to the set of positive training instances as T, the set of negative training instances as F, and assume that $T \cap F = \varnothing$. Each element of $T \cup F$ is a Boolean vector $\mathbf{x} = (x_1, \ldots, x_n)$, in which each x_i is referred to as a *feature* or an *attribute* and

$$x_i = \begin{cases} 1 & \text{if the } i^{th} \text{ feature is true,} \\ 0 & \text{if the } i^{th} \text{ feature is false.} \end{cases}$$

It is well known that a *Boolean function*, that is, a mapping $f : B^n \to B$, where $B = \{0, 1\}$, can be represented by a $2^n \times (n+1)$ table, with the first n columns representing a point in B^n and the $n+1$ column representing the value of the function at each point. Similarly, the set $T \cup F$ can be seen to represent a *partial Boolean function* (pBF), that is, a Boolean function in which the value at some points in B^n is undefined. We shall refer this pBF as f. An *extension*

BEST COPY AVAILABLE

of a pBF, is a (fully defined) Boolean function which "agrees" with the pBF at every point at which the pBF is defined. An important extension is the Boolean function f^+ which is defined as

$$f^+(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \notin F, \\ 0 & \text{if } \mathbf{x} \in F. \end{cases}$$

In general, machine learning methodologies can be viewed as solving three sub-problems:

> *Feature selection* involves identifying a "small" set of attributes or features which is sufficient for differentiating instances in T from those in F.
>
> *Rule generation* involves using selected features to create elementary Boolean conjunctions which can be used as "rules" to differentiate instances in T from those in F.
>
> *Rule selection* involves identifying the "best" rules and then combining them into a single rule which can be used to differentiate instances in T from those in F.

In LAD, these three sub-problems are called ***support set generation***, ***pattern generation*** and ***theory generation***. A set of features is a ***support set*** if the elements of T and F can be separated (i.e. distinguished) by using only the features in the set [3]. Therefore a support set is a set of essential features. A ***minimal*** support set is a support set, which does not contain any other support set.

A ***positive pattern*** [6] is a elementary Boolean conjunction C such that

1. $C(\mathbf{x}) = 0$ for every $\mathbf{x} \in F$
2. $C(\mathbf{x}) = 1$ for at least one vector $\mathbf{x} \in T$

A pattern is called a ***positive prime pattern*** if for every conjunction C' obtained by dropping a feature from C, there exists a vector $\mathbf{x} \in F$ such that $C'(\mathbf{x}) = 1$. It can be seen that every positive prime pattern is a prime implicant[1] of the extension f^+. The coverage of a positive pattern C is

$$\frac{|\{\mathbf{x} \in T : C(\mathbf{x}) = 1\}| + |F|}{|T \cup F|}$$

That is, it is the proportion of points in the training set which the positive pattern C correctly classifies. The above formula uses the fact that a positive pattern by definition will always correctly classify all points in F. For example, if $|T| = 10$, $|F| = 15$ and if a positive pattern correctly classifies 5 points in T, then its coverage is 20/25. It should be noted that the concepts for negative patterns are defined similarly by interchanging T and F in the discussion above.

---

[1] An elementary conjunction C is an ***implicant*** of a Boolean function f if $C(\mathbf{x}) \leq f(\mathbf{x})$ for all $\mathbf{x} \in B^n$, i.e. $C(\mathbf{x}) = 1 \Rightarrow f(\mathbf{x}) = 1$ for all $\mathbf{x} \in B^n$. An implicant, C is said to be ***prime*** if dropping any literal causes it not to be an implicant.

BEST COPY AVAILABLE

A *theory* is a Boolean function which agrees with each $\mathbf{x} \in T \cup F$. For example, the extension f^+ is a theory. It can be seen that a theory can be expressed as a *DNF* in which each term is a positive pattern [6].

Example [2]

Point	x_1	x_2	x_3	f
a	1	1	0	1
b	0	1	0	1
c	1	0	1	1
d	1	0	0	0
e	0	0	1	0
f	0	0	0	0

It can be seen that x_1, x_2 and x_3 form a minimal support set since points c and e differ only in x_1, points a and d differ only in x_2, and points c and d differ only in x_3. It can also be seen that the positive patterns are x_2 and x_1x_3 and that the negative patterns are $\bar{x}_1\bar{x}_2$, $\bar{x}_1x_3$ and $\bar{x}_2\bar{x}_3$. Finally, $x_2 \vee x_1x_3$ is a theory.

LAD seeks to find the optimal (many different criteria for optimality may be employed) support sets, patterns and theories via combinatorial optimization. For example, we may be interested in finding all minimal support sets, all prime patterns and all minimal (i.e. containing no redundant terms) theories. This approach, however, involves solving problems which are extremely computationally expensive. For example, finding all minimal support sets involves solving a Set Covering Problem (SCP) which is known to be NP-hard [7].

2 Implementing LAD

This section discusses an implementation of LAD which solves variants of the support set, pattern and theory generation problems discussed in Section 1. These variants can be solved in polynomial time and are therefore practical for use in solving large machine learning problems. This implementation was developed by one of the authors (Boros), in the Perl programming language and preparations are being made for making this code publicly available.

It should be mentioned that the actual training set which is presented to the LAD algorithm is a set of real vectors where each vector represents a document and each component of a vector represents the relative frequency of a term in that document. This document representation is prepared by running a Perl based indexer which does not do stemming, but does remove the stopwords specified in the Cornell SMART stopword list [1]. As will be seen, these real vectors will be converted to binary vectors during support generation.

We define the value of the Boolean variable x_i for document j as follows

---

[1] The Cornell SMART stopword list can be found at `ftp://ftp.cs.cornell.edu/pub/smart/english.stop`

BEST COPY AVAILABLE

$$x_{ij} = \begin{cases} 1 & \text{if } y_{ij} \geq t_i, \\ 0 & \text{if } y_{ij} < t_i. \end{cases}$$

where y_{ij} is the relative frequency, in the j^{th} document, of a term associated with variable x_i, and t_i is a scalar which is calculated by the support set generation algorithm.

The ***support set generation*** algorithm does not attempt to generate all minimal support sets, but instead employs a greedy heuristic which tries to find a set of binary variables x_i, of minimum cardinality, such that the minimum Hamming distance between any two vectors $a \in T$ and $b \in F$ is at least k, where k is a parameter of the algorithm. Increasing the value of k will result in more variables entering the support set, since it takes more variables to "push the sets farther apart". Sometimes it is not possible to separate T and F by a Hamming distance of at least k. In these cases we utilize a lexicographic rule to decide which variables enter the support set.

The ***pattern generation*** algorithm does not try to find all minimal patterns, but rather exhaustively generates all patterns which have a coverage of at least c and which have degree less than d, where c and d are parameters of the algorithm. Since the expected number of these patterns in randomly generated training sets is polynomial, this algorithm can be implemented to run in expected polynomial-time [5]. In cases where either no positive or no negative patterns were generated with the initial settings for c and d, the value of c was iteratively lowered until at least one positive and at least one negative pattern was generated.

While the Perl LAD package does support *theory generation*, we do not use this feature in our TREC-9 runs. Instead we use all the patterns to calculate a real-valued score σ as follows

$$\sigma = \sum_{i \in P} c(i) - \sum_{j \in N} c(j)$$

where P is a set of positive patterns, N is a set of negative patterns, and $c(i)$ denotes the coverage of the i^{th} pattern.

For the routing runs, we simply calculate σ for every document in the test set and present the first 1000 documents ranked by σ. For the adaptive filtering runs we use σ to create a linear classifier with the rule that a document in the test set is retrieved only if $\sigma > 0$. We did experiment with computing the threshold in a less arbitrary manner. Specifically, for a given topic, we computed σ for every document in the training set and then found a threshold value τ which correctly classifies the maximum number of training set documents using the following rule: documents for which $\sigma \geq \tau$ are classified as relevant and those for which $\sigma < \tau$ are classified as irrelevant. However, we found that simply letting $\tau = 0$ generally resulted in better performance.

3 Results

For both the routing and the adaptive filtering runs we set the parameters discussed in Section 2 as follows:

- desired Hamming distance between T and $F = 5$
- maximum degree of positive and negative patterns = 5
- minimum fraction of T covered by a positive pattern and minimum fraction of F covered by negative pattern = 0.95

This resulted in

- about 5 to 75 terms in the support set for each topic
- a large number of degree 1, 2 and 3 patterns even though the maximum degree is set to 4 or 5
- about 2 to 150 patterns generated for each topic

The ability to generate patterns with such a high coverage is uncommon when LAD is applied to data sets other than information retrieval ones. In addition, anecdotal evidence supports the idea that the support set is a short set of terms which are "relevant" to the topic at hand.

3.1 Routing Sub-Task

In the TREC-9 the routing sub-task we submitted two types of runs for the OHSU topics and the same two types of runs for the MESH-SAMPLE topics. The two types differed in that the antrpohsu00, antrpms00 runs only used the coverage of positive patterns in the computation of σ while the antrpnohsu00, antrpnms00 runs used the coverage of both positive and negative patterns. The following tables present score statistics for each run. We list the mean, standard deviation, maximum, median and minimum of our scores for each run. We also list the number of topics in which our score was greater or equal to the track median score, and the number of times we achieved the maximum score of the track.

Run: **Topic Name:** **Number Topics:**	antrpohsu00 OHSU 63
Mean	0.099
Standard Deviation	0.137
Max	0.648
Median	0.054
Min	0.000
# times $\geq$ median	6
# times = max	1

Run:	antrpnohsu00
Topic Name:	OHSU
Number Topics:	63
Mean	0.177
Standard Deviation	0.160
Max	0.690
Median	0.132
Min	0.000
# times ≥ median	20
# times = max	0

Run:	antrpms00
Topic Name:	MESH-SAMPLE
Number Topics:	500
Mean	0.078
Standard Deviation	0.156
Max	0.856
Median	0.004
Min	0.000
# times ≥ median	12
# times = max	1

Run:	antrpnms00
Topic Name:	MESH-SAMPLE
Number Topics:	500
Mean	0.158
Standard Deviation	0.198
Max	0.855
Median	0.065
Min	0.000
# times ≥ median	93
# times = max	12

3.2 Adaptive Filtering Sub-Task

In the TREC-9 the adaptive filtering sub-task we submitted two runs for the OHSU topics. Both the antadapt001 and antadapt002 runs used ten copies of the topic as positive training documents in addition to the two initial training documents provided. The antadapt001 and antadapt002 differed in that antadapt002 added some randomly selected documents from the training set which to act as negative training documents.

The adaptive strategy employed was as follows:

- a document is retrieved only when $\sigma > 0$
- all retrieved documents are added to the training set

- support set and pattern generation are only rerun when we "make a mistake", that is, when an irrelevant document is retrieved

Run:	antadapt001
Topic Name:	OHSU
Number Topics:	63
Measure:	T9P
Mean	0.088
Standard Deviation	0.132
Max	0.580
Median	0.040
Min	0.000
# times ≥ median	8
# times = max	1

Run:	antadapt002
Topic Name:	OHSU
Number Topics:	63
Measure:	T9P
Mean	0.102
Standard Deviation	0.150
Max	0.791
Median	0.056
Min	0.000
# times ≥ median	9

Run:	antadapt001
Topic Name:	OHSU
Number Topics:	63
Measure:	T9U
Mean	-32.270
Standard Deviation	45.406
Max	50.00
Median	-9.00
Min	-100.00
# times ≥ median	26
# times = max	6

Run:	antadapt002
Topic Name:	OHSU
Number Topics:	63
Measure:	T9U
hline Mean	-43.571
Standard Deviation	53.405
Max	118.00
Median	-22.00
Min	-100.00
# times $\geq$ median	24
# times = max	4

4 Conclusion

Our best performance in both the routing and the adaptive filtering sub-tasks was substantially below the hypothetical "median system". In our best routing run on the OHSU topic set, we achieved the median about one-third of the time, while on the MESH-SAMPLE topic set we only achieve the median about one-fifth of the time. Although we did not tune our algorithm for any particular utility function, it appears that that it is more nearly tuned to T9U than T9P. Using T9U, we achieved the median about 38% of the time on the OHSU topic set. It is also interesting to note that in the adaptive filtering run, the addition of randomly selected documents for use as a negative training set did not substantially alter the performance. Our results in TREC-9 imply that more experimentation on how to utilize LAD for information retrieval is required. One conclusion is that using both positive and negative patterns resulted in better performance.

Ideas which might worth considering include

1. the use of support set terms for query expansion
2. incorporating stemming into our indexer
3. utilization of other term weighting schemes
4. development of additional methods for utilizing patterns in a document scoring function
5. development of more intelligent calculations of the threshold τ used in the linear classifier used in the adaptive filtering runs

References

[1] Endre Boros, Peter L. Hammer, Toshihide Ibaraki, and Alexander Kogan. Logical analysis of numerical data. *Mathematical Programming*, 79:163–190, 1997.

[2] Endre Boros, Peter L. Hammer, Toshihide Ibaraki, Alexander Kogan, Eddy Mayoraz, and Ilya Muchnik. An implementation of logical analysis of data. *IEEE Transactions on Knowledge and Data Engineering*, in press.

[3] Endre Boros, Takashi Horiyama, Toshihide Ibaraki, Kazuhisa Makino, and Mutsunori Yagiura. Finding small sets of essential attributes in binary data. *To appear in the Second International Conference on Intelligent Data Engineering and Automated Learning*, Hong Kong, December 13-15, 2000.

[4] Endre Boros, Paul B. Kantor, and Dave J. Neu. Pheromonic representation of user quests by digital structures. *In Proceedings of the 62nd Annual Meeting of the American Society for Information Science*, 36:633–642, 1999.

[5] Endre Boros, Lijie Shi, and Mutsunori Yagiura. Generating all good patterns in polynomial expected time. *To appear in the 6th International Symposium of Artificial Intelligence and Mathematics, Fort Lauderdale, FL*, pages 633–642, January 5-7, 2000.

[6] Yves Crama, Peter L. Hammer, and Toshihide Ibaraki. Cause-effect relationships and partially defined boolean functions. *Annals of Operations Research*, 16:299–326, 1988.

[7] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeeman and Company, 1979.

[8] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[9] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufamnn, 1993.

Sabir Research at TREC 9

Chris Buckley and Janet Walz

Sabir Research participated in TREC-9 in a somewhat lower key fashion than normal. We participated only in the Main Web Task, and in the Query Track. Most of our interesting work and analysis was done in the Query Track, and is reported in the TREC-9 Query Track Overview. Here we report very briefly on the Main Web Task; briefly because there really isn't much interesting to say this year!

We used the SMART Information Retrieval System Version 13.3.6 for our runs. SMART was developed at Cornell and continues to be developed at Sabir Research. The basic algorithms have been described numerous times in our past TREC papers [1, 2, 3]; we made no major changes this year. They includes blind feedback with query zoning, and looking at correlated terms.

The Web data itself posed no problems for SMART. This was our first Web test collection, but basic indexing and retrieval was straightforward (modulo a forgotten check to ensure no single word exceeded 512 characters in length.) SMART indexes at 3 to 4 GBytes per hour on a cheap PC running Linux.

What does pose a problem is trying to take advantage of the additional Web information available. In our retrospective tests on last year's TREC-8 Web data, and in our tests both before and after TREC-9 submissions, nothing we tried seemed to affect the results much! Experiments with anchor text, links, and trying to emphasize certain parts of the documents all had basically no effect on retrieval results. In most cases they had a minor detrimental effect. Even basic retrieval and indexing variations such as stemming, phrasing, and document length normalization had little effect on the Web results; less effect than we would have expected. Given our results are 10% - 20% under the current top groups, which include other groups that were running SMART like AT&T [4], we obviously need to look at things in more detail.

One known weakness in our current setup is choice of query expansion terms from blind feedback. We haven't yet played around with options here because we have an investigation in the area planned for the near future as we make major changes to SMART. SMART currently offers several choices for expansion, but Sabir has stayed with expanding by terms related to as many top documents as possible. That appears to be non-optimal for recent TREC test collections, as too many expansion terms are not content-bearing terms. In earlier TRECs, with more relevant and near relevant documents per query, we were able to pull in the general terms which described the query content area. We need some method of distinguishing which queries we can draw in these good general

BEST COPY AVAILABLE

Run	Mean Avg Prec	R-Prec	NumRelRet
Sab9web1	.1265	.1518	1250
Sab9web2	.2122	.2463	1468
Sab9web3	.2159	.2464	1456
Sab9web4	.2091	.2485	1476
Sab9web5	.2018	.2400	1468

Table 1: TREC-9 Main Web Task Results

terms, and which queries we need to target specific terms.

We submitted 5 runs to the Main Web Task. Sab9web1 was run with the title words only from the topic statement. Sab9web2, Sab9web3, and Sab9web4 used the entire topic statement. Sab9web5 used the entire topic statement plus used link information from the Web pages. As can be seen from Table 1, as expected Sab9web1 is significantly worse than the others, but all the variations we tried using the entire topic statement didn't result in any changes. All of our runs are above average for their respective categories, but not in the top group of systems this year.

In conclusion, Sabir Research participated in the Main Web Task and the Query Track. The Query Track investigation is reported elsewhere. In the Web Task, we used the same basic approach as the past few years and got above average, but not top results. We were unable to get any improvement using Web-specific data such as links, anchor texts, and content placement. That doesn't mean the data may not be useful in the future, only that we were unable to take advantage of it here.

References

[1] Chris Buckley, Mandar Mitra, Janet Walz, and Claire Cardie. Using clustering and SuperConcepts within SMART: TREC-6. In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Sixth Text REtrieval Conference (TREC-6)*, pages 107–124, August 1998. NIST Special Publication 500-240. Electronic version available at http://trec.nist.gov/pubs.html.

[2] Chris Buckley, Mandar Mitra, Janet Walz, and Claire Cardie. SMART high precision: TREC 7. In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*, pages 285–298, August 1999. NIST Special Publication 500-242. Electronic version available at http://trec.nist.gov/pubs.html.

[3] Chris Buckley and Janet Walz. SMART in TREC 8. In Voorhees and Harman [5]. NIST Special Publication 500-246. Electronic version available at http://trec.nist.gov/pubs.html.

[4] Amit Singhal, Steve Abney, Michiel Bacciani, Michael Collins, Donald Hindle, and Fernando Pereira. AT&T at TREC-8. In Voorhees and Harman [5]. NIST Special Publication 500-246. Electronic version available at http://trec.nist.gov/pubs.html.

[5] E.M. Voorhees and D.K. Harman, editors. *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, 2000. NIST Special Publication 500-246. Electronic version available at http://trec.nist.gov/pubs.html.

QUERY EXPANSION SEEN THROUGH RETURN ORDER OF RELEVANT DOCUMENTS

Walter Liggett
National Institute of Standards and Technology
Gaithersburg, MD 20899 USA
walter.liggett@nist.gov

Chris Buckley
SabIR Research, Inc.
Gaithersburg, MD 20878 USA
chrisb@sabir.com

Abstract

There is a reservoir of knowledge in data from the TREC evaluations that analysis of precision and recall leaves untapped. This knowledge leads to better understanding of query expansion as this paper demonstrates. In many TREC tasks, the system response required is an ordered list of 1000 document identifiers. Instead of just using the identifiers to determine the positions of relevant documents in each list, we extract from each list the identifiers of the relevant documents and compare document ordering in these sub-lists. In other words, we consider the return order of relevant documents. We use Spearman's coefficient of rank correlation to compare sub-lists and multidimensional scaling to display the comparisons. Applying this methodology to data from the TREC Query Track, specifically, to system responses to twenty restatements of each of four topics, we show how two systems with query expansion differ from four systems without. We observe return-order variations caused by topic restatement and determine how query expansion affects these variations. For some topics, query expansion reduces the sizes of these variations considerably.

1. INTRODUCTION

Progress in information retrieval (IR) depends on understanding how search results vary with IR system inputs, in particular, the topic (the information need) and the query (the natural language statement that conveys the topic to the IR system). In pursuit of this understanding, TREC evaluations of IR systems elicit system responses (the TREC 1000-document lists) to a variety of topics and for each topic, a variety of queries. TREC data include responses from IR systems with different features and thus, reflect the dependence of feature-related response differences on system inputs. Understanding this dependence can lead to better IR systems. This paper introduces an approach to studying this dependence and applies this approach to comparison of IR systems with and without query expansion.

Computing a performance value from each system response is the customary first step in a TREC analysis. The basis for this is a defining statement of the topic which an assessor then uses to designate some of the documents in the collection as relevant. The document identifiers in the ordered list show the positions in the list occupied by relevant documents. These positions are all that is needed to compute precision and recall measures of performance. In fact, because there is no designation of degree of relevance for documents in the collection beyond the relevant-irrelevant dichotomy, a performance measure cannot depend on anything except these positions. However, it is not necessary to start with performance values in the analysis of TREC data. Performance values are convenient in that they allow averaging for summarization. Nevertheless, when studying the dependence of system responses on inputs, limiting one's options by insisting on use of a performance-based analysis may hamper the study.

New avenues for analysis are opened when one allows, as the first step, computation of a measure of dissimilarity for each pair of system responses (Banks, et al., 1999). A dissimilarity-based analysis does not preclude a performance-based analysis since the absolute difference between two performance values is a dissimilarity. However, a dissimilarity measure can be computed from the order in which particular documents occur in either the entire list or in the relevant-document part of the list. For example, as detailed in the next section, one can eliminate the irrelevant documents from each list so that one has ordered lists of relevant documents and then compute dissimilarities from these reduced lists by taking into account the actual identifiers in the lists. Such a dissimilarity can be said to depend on the return order of relevant documents. Because such a dissimilarity does not depend on the positions of irrelevant documents in the original list, it is clear that this dissimilarity may reflect aspects of the TREC data that are not portrayed by precision and recall measures.

We apply our dissimilarity-based analysis to the Query Track data of TREC-8 and TREC-9 for the purpose of studying query expansion (Buckley and Walz, 2000). The Query Track data are unique in that they consist of system responses to several restatements of each of 50 topics. As implemented in a well-regarded class of information retrieval systems, the response is computed by first deriving a weighted set of terms (key words) from the original statement of the topic and then matching this query set against the terms in the document collection. The first step may involve a procedure that adds terms to the original query set by examining documents judged particularly relevant in an initial search of the document collection. This procedure is intended to uncover terms pertinent to the need for information that are not in the original statement of this need. Selecting additional terms for the query set may be done by the user in which case the procedure is called relevance feedback (Berry and Browne, 1999) or may be done automatically in which case the procedure is called blind feedback or (automatic) query expansion. Because they include systems with query expansion, the Query Track data allow us to see how systems with and without query expansion handle restatements of information needs.

In thinking about the performance of query expansion, one might suppose that when a system with more effective query expansion is applied to alternative statements of the same need for information, the lists returned would vary less. Query expansion that leads to less variation might be seen in two ways. First, such query expansion should improve performance in terms of precision and recall by bringing to the fore relevant documents that do not include the same terminology as the original query. Second, such query expansion should make document order in the relevant-document subsequences less dependent on the particular terminology used in the query. In terms of a dissimilarity measure that reflects the ordering of relevant documents, query expansion should reduce the dissimilarities among responses to alternative statements of the same information need. In this paper, we pursue this second manifestation.

Discussion of the approach introduced in this paper begins in the next section with specification of the dissimilarity measure. Analysis of the dissimilarities thus computed requires multidimensional scaling for graphical presentation as illustrated in Section 3. Finally, future work needed to take full advantage of the approach is discussed in Section 4.

2. DISSIMILARITY MEASURE

The dissimilarity measure used in this paper leads to fresh insights from the TREC 1000-document lists. Since two such lists can be compared in many ways, there are alternative measures. It is a contribution that we have found an effective measure although it may not be the most effective. We begin this section by specifying our dissimilarity measure and then review alternatives.

For each TREC topic, there is available a set of documents that assessors have determined to be relevant to the topic. For the topic under consideration, let there be n_R documents in this set, and let these documents be indexed by $i = 1, \dots, n_R$. In a TREC 1000-document list, let n be the number of relevant documents returned, and if relevant document i is returned, let r_i $(1 \le r_i \le 1000)$ denote its position in the list.

As the basis of our dissimilarity measure, we let R_i denote the position of document i in what is left of the list when the irrelevant documents have been removed. In other words, we let R_i denote the rank of r_i among the positions of the relevant documents, $r_1, \dots, r_n$. Thus, if $r_i = \min(r_1, \dots, r_n)$, then $R_i = 1$, that is , document i is returned first among the relevant documents. To the relevant documents not returned, we assign the same R_i, the average of ranks $n + 1$ to n_R. Thus, if relevant document i is not returned, we let $R_i = (n_R + n + 1)/2$. Our dissimilarity measure is based on the R_i thus defined. Note that irrelevant documents positioned in different ways in a 1000-document list can lead to the same $R_1, \dots, R_{n_R}$. The irrelevant documents influence our dissimilarity measure only through n, the number of relevant documents returned.

Our dissimilarity measure is obtained from Spearman's coefficient of rank correlation (Gibbons, 1985). Consider two 1000-document lists with $n^{(p)}$ relevant documents returned in the first and $n^{(q)}$ in the second and with relevant document return order $R_1^{(p)}, \dots, R_{n_R}^{(p)}$ for the first and with $R_1^{(q)}, \dots, R_{n_R}^{(q)}$ for the second. Spearman's coefficient of rank correlation adjusted for the relevant documents not returned is given by

$$s_{pq} = \frac{n_R^3 - n_R - 6\sum_{i=1}^{n_R} (R_i^{(p)} - R_i^{(q)})^2 - (U^{(p)} + U^{(q)})/2}{\sqrt{[n_R^3 - n_R - U^{(p)}][n_R^3 - n_R - U^{(q)}]}},$$

where

$$U^{(p)} = (n_R - n^{(p)})^3 - (n_R - n^{(p)})$$

$$U^{(q)} = (n_R - n^{(q)})^3 - (n_R - n^{(q)}).$$

Converting s_{pq}, which is a similarity measure, to our dissimilarity measure, we obtain

$$\delta_{pq} = \sqrt{1 - s_{pq}}.$$

To develop an understanding of this dissimilarity measure, one might consider the case in which all the relevant documents are returned in both lists. In this case, s_{pq} is just the product-moment correlation coefficient computed from the ranks $R_1^{(p)},\ldots, R_{n_R}^{(p)}$ and $R_1^{(q)},\ldots, R_{n_R}^{(q)}$. Moreover, δ_{pq} is proportional to

$$\sqrt{\sum_{i=1}^{n_R} (R_i^{(p)} - R_i^{(q)})^2}.$$

The contribution of relevant document i to this quantity is the difference $R_i^{(p)} - R_i^{(q)}$, the difference between the two lists in the relevant-document rank of document i. Thus, δ_{pq} measures dissimilarity in terms of the relevant documents at the fore in each list.

To put our dissimilarity measure in context, we consider its relation to performance measures and to other dissimilarity measures. Because the dissimilarity between two lists can be defined as the absolute value of the difference between the performance measures for the two lists, one can see how to turn a performance measure approach into a dissimilarity approach. The reverse is not generally possible in the sense that one usually cannot find a univariate performance measure that gives the dissimilarities among several lists. This is not surprising since one would expect that description of the differences among 1000-document lists would require many dimensions.

A popular performance measure is average precision. One can calculate it by first sorting the relevant document positions $r_1,\ldots, r_n$ to obtain $r_{(1)},\ldots, r_{(n)}$, where $r_{(1)} \le r_{(2)} \le \ldots \le r_{(n)}$ and then computing

$$\frac{1}{n_R}\sum_{i=1}^{n} \frac{i}{r_{(i)}}.$$

Note first that this performance measure involves only the distinction between relevant and irrelevant documents and nothing else derived from the document identifiers. Other performance measures based one way or another on precision and recall have this same property. It is moreover true that because the documents in the collection are not graded according to relevance, there is no way to define a performance measure that involves more than the relevant-irrelevant distinction. This immediately shows that our dissimilarity measure involves a novel aspect of system responses. Performance measures involving precision and recall are generally measures of how well irrelevant documents are rejected. On the other hand, our dissimilarity measure makes different use of the document identifiers and thereby opens up the possibility of new insights from TREC data.

There are dissimilarity measures other than the one specified in this paper and those based on performance measures. One might invent a dissimilarity measure that reflects the difference between the irrelevant documents in two lists. One might compare the return order of relevant documents by computing Kendall's tau instead of Spearman's coefficient of rank correlation. It is possible that use of a few more dissimilarity measures would give further insight into TREC data.

3. DISPLAY BY MULTIDIMENSIONAL SCALING

Say that one wants to compare a group of system responses (1000-document lists) and that one computes a dissimilarity for each pair in the group and thus the dissimilarity matrix for the group. As argued in Section 2, this might lead to insights that cannot be obtained from any performance measure. However, looking at the dissimilarity matrix is unlikely to produce much insight. Rather, insight can be obtained from a dissimilarity matrix through multidimensional scaling (Cox and Cox, 1994; Kruskal and Wish, 1978; Rorvig, 1999). This technique produces points on a plane, one point for each system response, arranged so that the Euclidean distances between the points approximate the dissimilarities. Thus, one obtains the system responses laid out in a two-dimensional configuration with more dissimilar responses farther apart. The configuration can then be further interpreted. The multidimensional scaling algorithm we use is Kruskal's isotonic multidimensional scaling, which is named "isoMDS" by Venables and Ripley (1999).

In this paper, we consider four topics that are interesting in themselves and illustrate the kinds of results one can obtain. For each topic, we compare six systems in terms of their responses to twenty queries. Thus, for each topic, we apply multidimensional scaling to a 120 by 120 dissimilarity matrix. The six systems are "IN7a," which is a version of the INQUERY system from the University of Massachusetts; "Saba," which is a version of the SMART system from SabIR Research; "humA," which is a Hummingbird system; "ok9u," which is a version of the Okapi system from Microsoft; "IN7e," which is another version of the INQUERY system; and "Sabe," which is another version of the SMART system. The first four systems do not employ query expansion whereas the last two do. (Further description of these systems is found elsewhere in this publication.) The twenty queries for each topic are, after removal of duplicates, the best performing according to a criterion based on recall-at-1000 values (given by n/n_R) for the six systems. Our criterion is a weighted combination of the six recall values with weights for the expansion systems twice as large because the number of expansion systems considered is half the number of non-expansion systems. The use of exactly this criterion is not essential to the results in this paper.

Multidimensional scaling gives a plot with a point for each query-system combination. As our plotting symbols, we combine a query symbol with a system symbol. The query symbols for the 21 queries carried over from TREC-8 are A, B, ..., U, respectively; and the query symbols for the 22 queries new in TREC-9 are v, w, a, b, ..., t, respectively. The system symbols are 1, 2, 3, 4, *, and #, respectively.

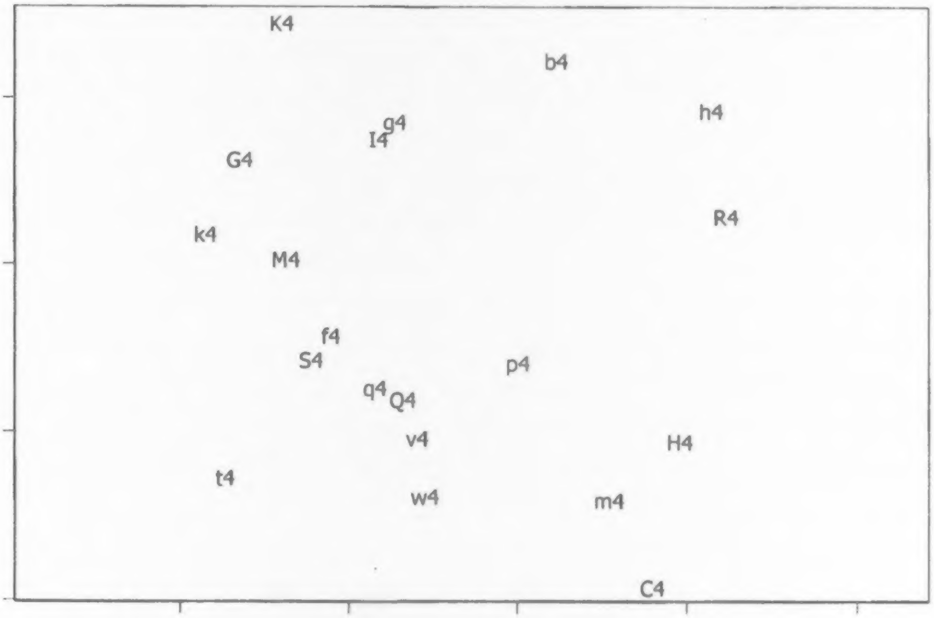


Figure 1. System "ok9u" Points for Topic 100.

Our first example is multidimensional scaling for topic 100. We do not begin by showing all 120 responses, however. In Figure 1, we show only the 20 points that correspond to the system "ok9u." We have blanked out the points corresponding to the other systems. Each point in this figure corresponds to a query. The meaning in this figure is obtained from relative distances among points. For example, we see that R4 is closer to h4 than to t4. In other words, R4 and h4 are less dissimilar than R4 and t4 or h4 and t4. Looking at the queries, we see that this is reasonable since R4 is "cocom control export," h4 is "America enforcing the terms of the COCOM agreement," and t4 is "policy, regulation or control of high technology transfer." Essentially, R4 and h4 are closer together because they share the term "cocom." Because Figure 1 shows only relative distances, the configuration of points can be shifted, scaled up or down by the same amount on each axis, and rotated without affecting the meaning. This is the reason why no values are attached to the axis tick marks.

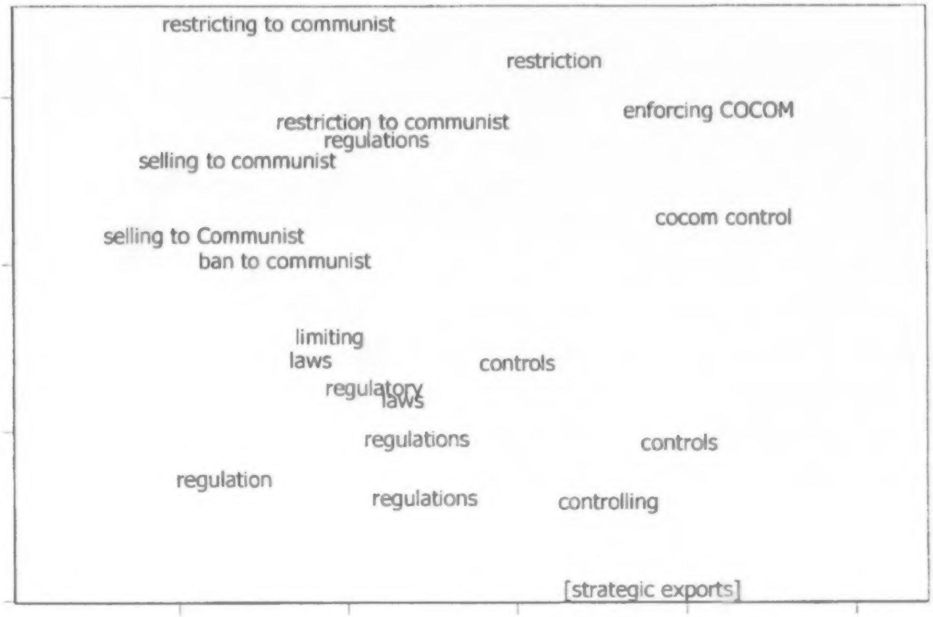


Figure 2. Selected Query Words at the "ok9u" Points for Topic 100.

Figure 2 is a somewhat subjective association of query texts with all the points in Figure 1. Because space on the figure prohibits printing the entire texts, we have selected a few words from each query. The word "export" appears in all the queries so we have omitted this word except in the one case in which the entire query is "strategic exports." What we see in Figure 2 are two axes that give meaning to the configuration. Horizontally, we see that the queries vary from reference to laws and regulations on the left to reference to control on the right. Vertically, we see that the queries vary from reference to general threats on the bottom to communist threats on the top. Thus, we see the variations in query wording that lead to the major differences in the response of the "ok9u" system.

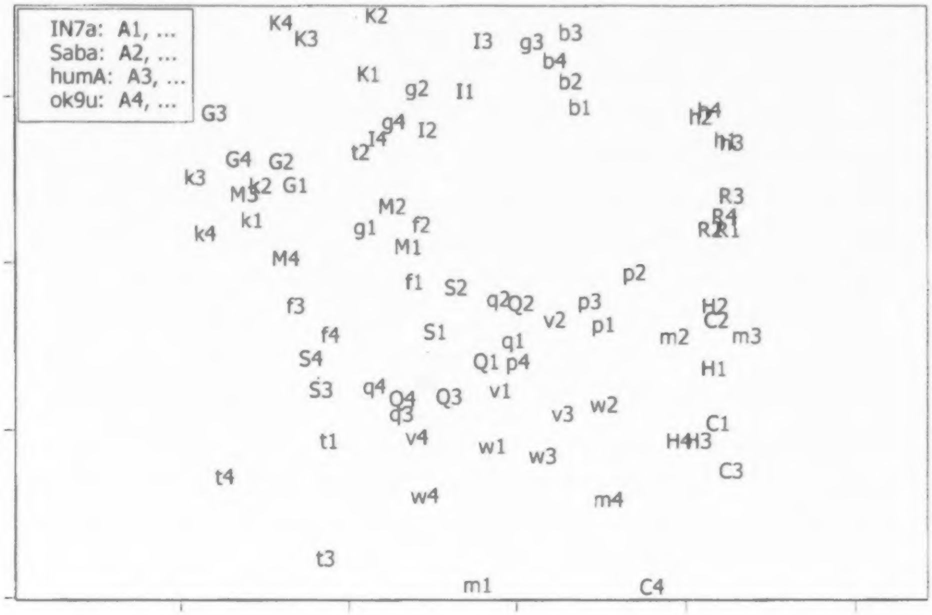


Figure 3. Topic 100 Points for the Non-Expansion Systems.

For topic 100, these query-wording axes hold for all the systems that do not employ query expansion. Figure 3 shows this. For a particular letter, the points with numbers 1, 2, 3, and 4 are generally close together. There are exceptions such as the distance t2 is from t1, t3, and t4. Nevertheless, if we were to associate query text with points for systems "IN7a," "Saba," or "humA," the resulting figures would be much like Figure 3. Thus, for this topic, the queries provide system-independent meaning to the space created by multidimensional scaling.

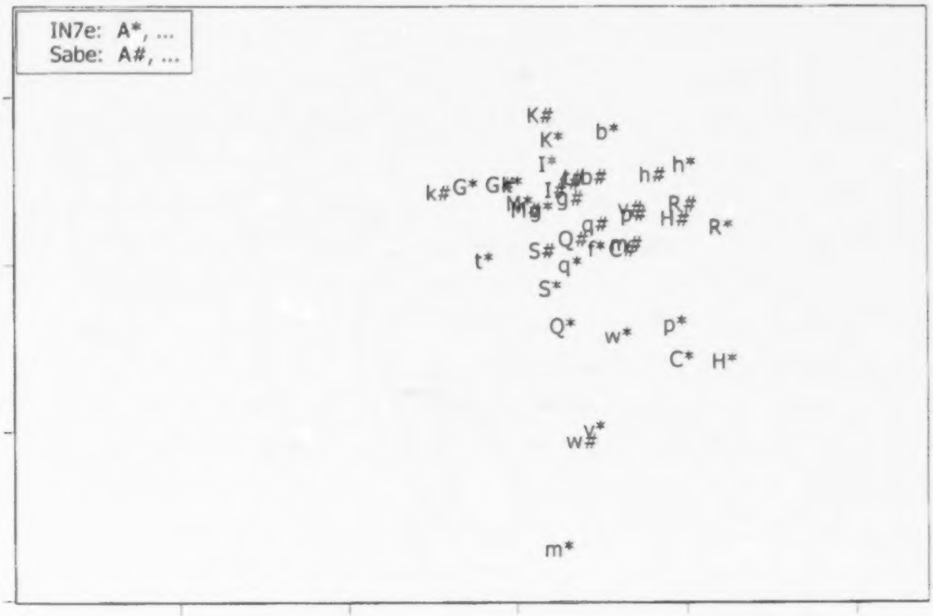


Figure 4. Topic 100 Points for Systems with Query Expansion.

Figure 4 shows that for Topic 100, query expansion is effective in the sense that it reduces the variation in system response due to query-to-query variation. In comparison of Figure 4 with Figure 3 (which both have the same scale), we see that the scatter in the responses of systems "IN7e" and "Sabe" is much less. Thus, query expansion as incorporated in these two systems makes the system response less dependent on the particular words chosen to express the topic, the need for information. In particular, there is less dependence on whether the query uses "control" or "regulation" and whether or not the query includes the term "communist."

One might question the point in Figure 4 labeled "m*." The response to this query is apart from the other responses in this figure. This point is the response of the system "IN7e" to the query "U.S.'s controlling of international exports." Given this query, this system was unable to retrieve documents that were retrieved by system "Sabe" with this query and documents retrieved by both systems with other queries. Noting an occurrence such as this could lead to insight into how a system can be improved.

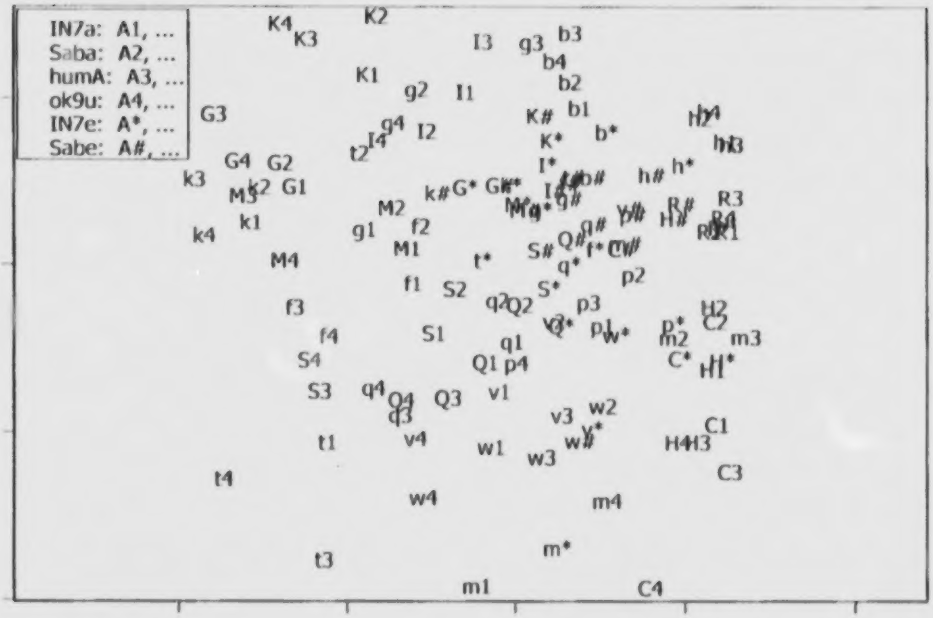


Figure 5. Multidimensional Scaling for Topic 100, All Points.

Figure 5 shows the configuration given by all 120 responses. Note that Figures 1-4 are all based on this configuration, that is, Figures 1-4 each exhibits only some of the 120 points but these at the locations shown in Figure 5. This figure is the one that actually gives the multidimensional scaling result that we use to interpret Topic 100. This figure gives an overview of all six systems. With the introduction provided by Figures 1-4, this overview might be helpful. As a place to start the analysis of a topic, a figure such as Figure 5 may require considerable effort before a reasonably complete interpretation can be obtained.

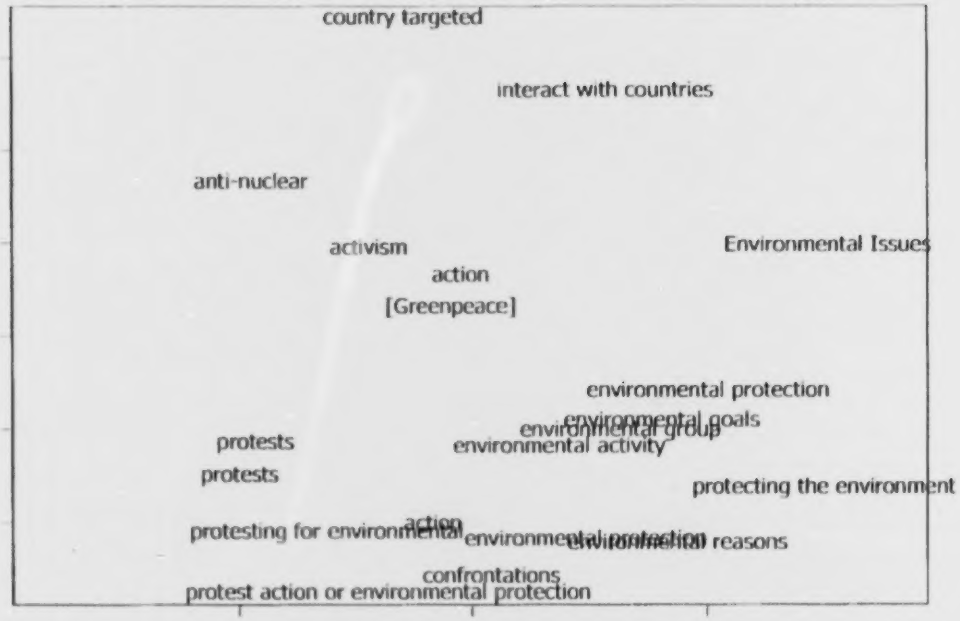


Figure 6. Selected Query Words at "ok9u" Points for Topic 78.

In our presentation of Topic 78, we begin with terms from the queries positioned at the points given by the system "ok9u" as shown in Figure 6. Because the 20 queries we consider all include the term "Greenpeace," we have omitted this term except in the case of the query that consists of the single word "Greenpeace." One way to interpret Figure 6 is to regard the horizontal axis as distinguishing Greenpeace regarded as a protest organization on the left and Greenpeace regarded as an environmental organization on the right. A proper interpretation of the vertical axis is less clear. Maybe the vertical axis distinguishes queries that refer to the actions Greenpeace takes from queries that refer to the targets of Greenpeace actions.

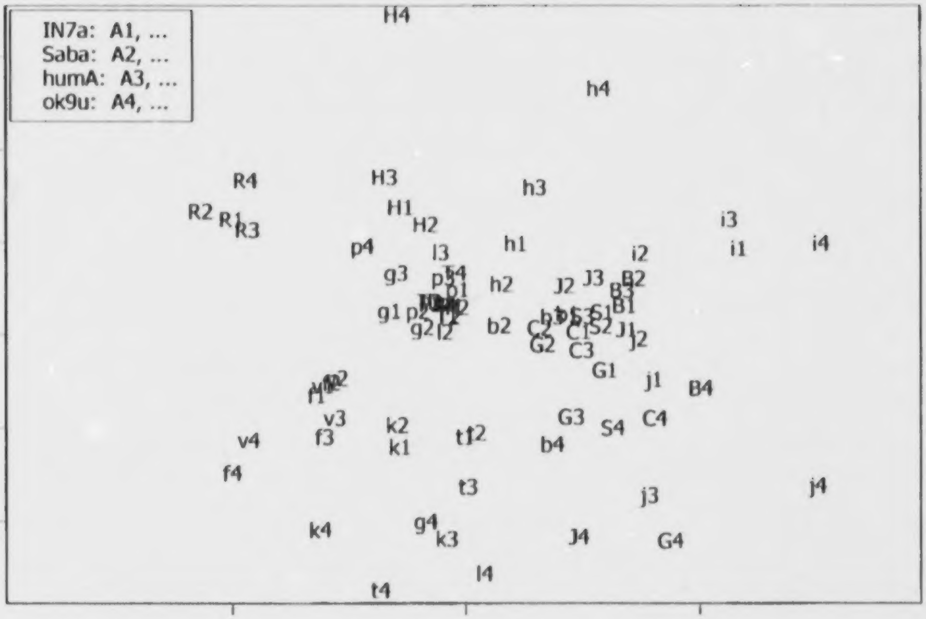


Figure 7. Topic 78 Points for Non-Expansion Systems.

Figure 7 shows that the most influential query terms affect the other non-expansion systems, "IN7a," "Saba," and "humA," as they do "ok9u." Generally, for each query, the four points for these four two systems lie close to each other. There are some exceptions such as the points g4, l4, and J4. Nonetheless, variation over the space portrayed by multidimensional scaling has meaning beyond the response of a particular system. This is the same observation that we made about topic 100 in conjunction with Figure 3.

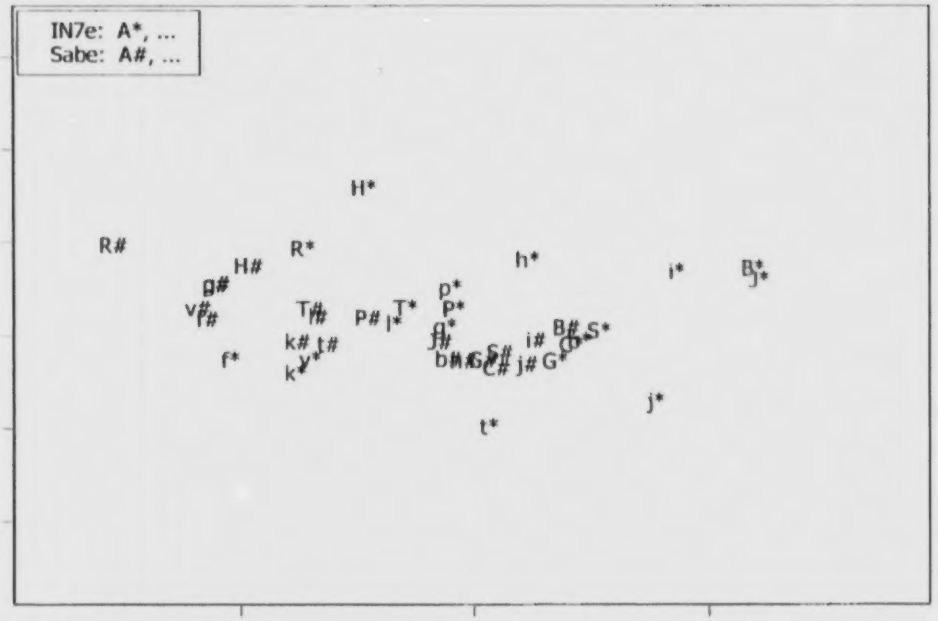


Figure 8. Topic 78 Points for Systems with Query Expansion.

Figure 8 shows the part of the configuration for Topic 78 produced by the systems with query expansion. Here, compared to Figure 7, we see less scatter in the vertical direction. Variation in the horizontal direction seems to have two properties. First, we see that for a query, the point for the system "Sabe" generally lies to the left of the point for the system "IN7e." One might say that the system "Sabe" tends to regard Greenpeace as a protest organization and that the system "IN7e" tends to regard Greenpeace as an environmental organization. What characteristics of the query expansion algorithms this reflects is an interesting question. Second, it seems that each system reduces the scatter in the horizontal direction but that this reduction is not toward the same point on the "protest" - "environmental" axis.

Because what can be learned has largely been shown in Figures 7 and 8, we omit the figure showing the entire configuration for Topic 78. This omitted figure does provide a better basis than Figures 7 and 8 for observing that the tendency of "Sabe" to regard Greenpeace as a protest organization is true with respect to the non-expansion systems as well as "IN7e."

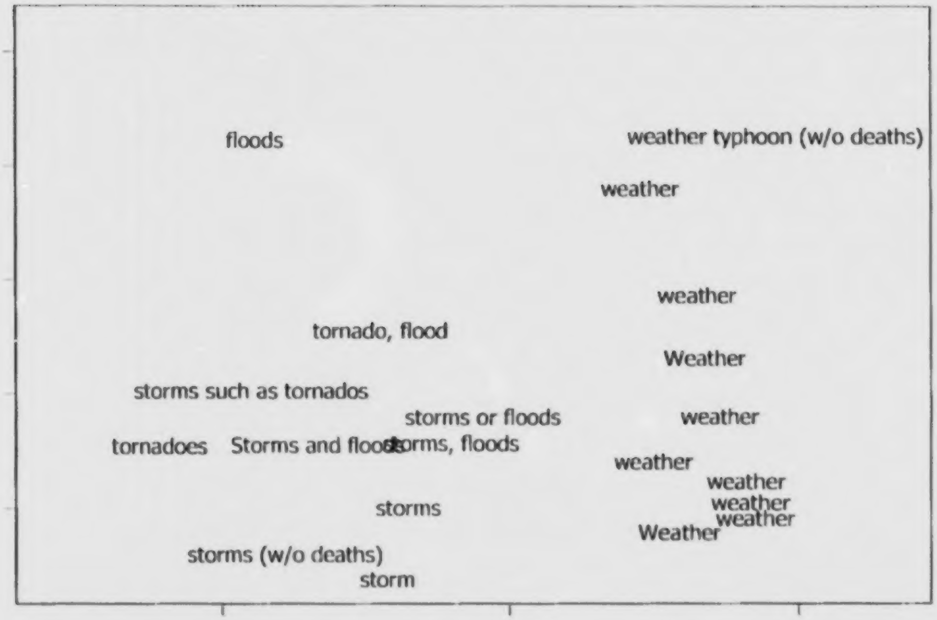


Figure 9. Selected Query Words at "ok9u" Points for Topic 59.

In our presentation of Topic 59, we again begin with terms from the queries positioned at the points given by the system "ok9u." These terms, which are shown in Figure 9, do not include the word "deaths" because this term occurs in almost all queries. Rather, we have indicated queries that do not include the word "deaths." In the horizontal direction, Figure 9 shows a clear separation between "storms" and "weather." The phrase "storm-related deaths" is equivalent to the phrase "weather-related deaths." Yet, as we will see, all of the systems respond as though these two phrases have different meaning.

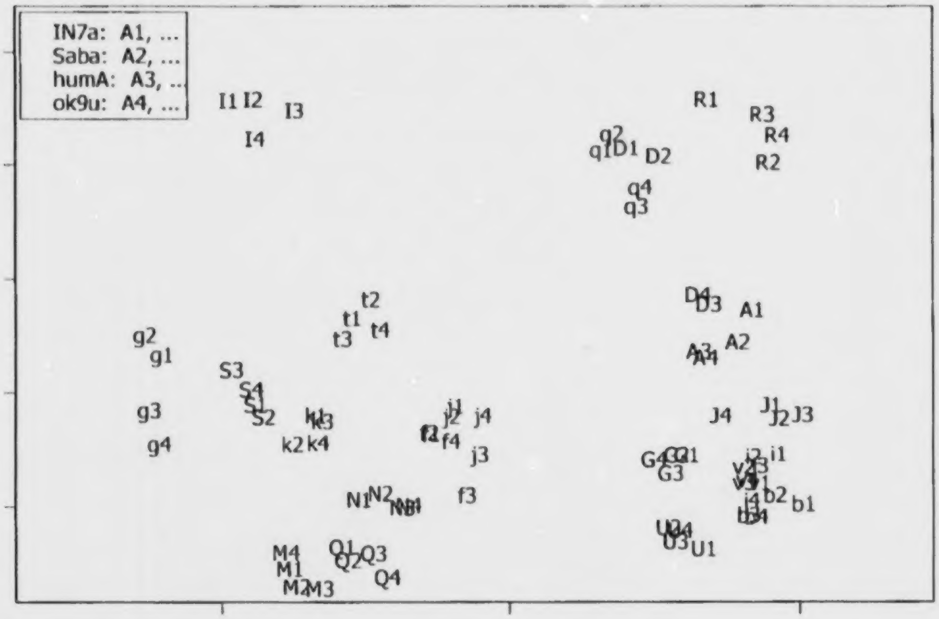


Figure 10. Topic 59 Points for Non-Expansion Systems.

Figure 10 shows the part of the configuration given by the non-expansion systems. These four systems respond similarly to each query. The scatter in this figure is largely query related, not system related. We see that the relative locations of query terms shown in Figure 9 apply to the other non-expansion systems as well.

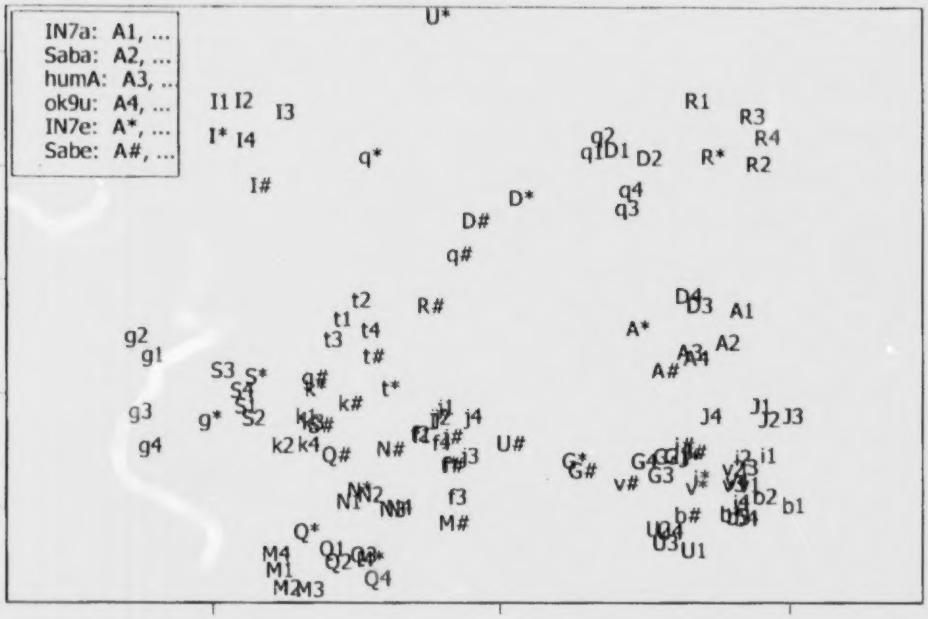


Figure 11. Multidimensional Scaling for Topic 59, All Points.

Figure 11 shows all 120 points in the topic 59 configuration. We see that although the query expansion systems move points associated with some queries, neither expansion system offers much reduction in the query-to-query scatter. Moreover, the "storm-related" - "weather-related" dichotomy also exists for these systems. One could draw a line just to the left of points U#, R#, and q* which would separate these two categories for all six systems. Thus, Topic 59 provides an example of failure of query expansion, failure to associate "storm-related" and "weather-related."

BEST COPY AVAILABLE

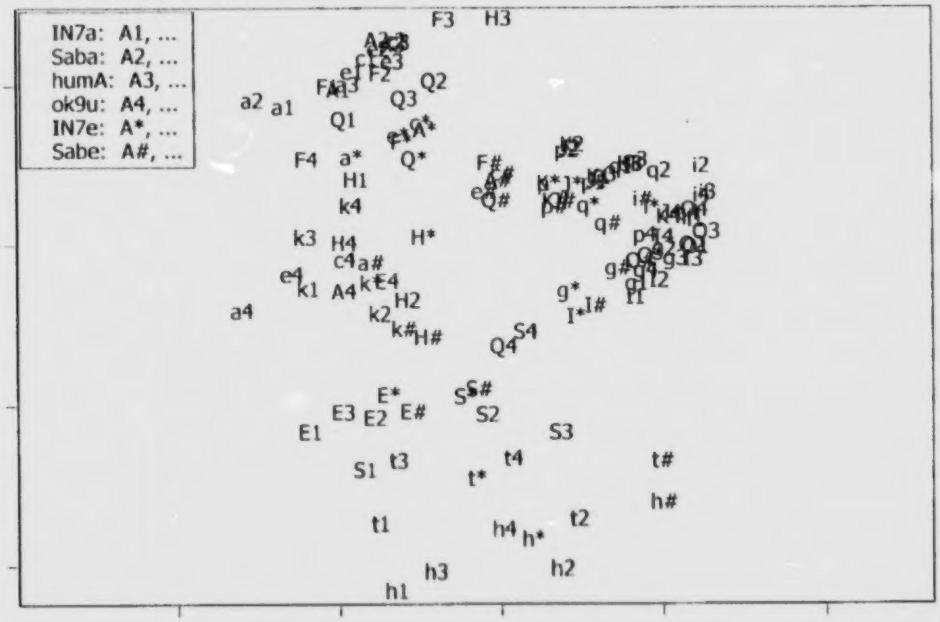


Figure 12. Multidimensional Scaling for Topic 86, All Points.

Results for Topic 86 cannot be summarized in the same way as the other topics discussed above. We begin with the entire configuration, which is shown in Figure 12. Study of this figure shows that whereas for topic 59, query differences are generally much greater than system differences, the situation for topic 86 is less clear. To show this, we compare the responses of "ok9u" with those of "Sabe."

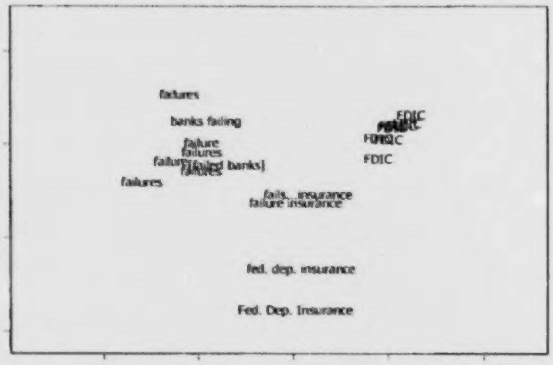


Figure 13. Selected Query Words at "ok9u" Points for Topic 86.

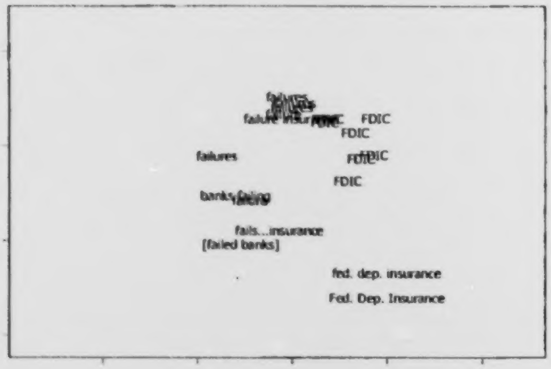


Figure 14. Selected Query Words at "Sabe" Points for Topic 86.

Figure 13 shows terms from each query at the "ok9u" locations. Consider the queries in three groups, those containing the term "FDIC," those containing the phrase "Federal Deposit Insurance Corporation," and those that do not refer to the FDIC in either way. All the queries in this third group contain the word "bank." Figure 13 shows that "ok9u" does not associate "FDIC" with the phrase spelled out although it should. In fact, none of the six systems do. The queries with "FDIC" form a tight cluster. The other queries are more spread out.

BEST COPY AVAILABLE

Figure 14 shows the same query terms as Figure 13 but at the "Sabe" locations. We see that "Sabe," an expansion system, brought some but not all of the queries in the third group closer to the "FDIC" queries. One would guess that this is caused by other terms in the queries but which terms is not clear. For topic 86, variation across the space defined by multidimensional scaling involves both query effects and system effects. Thus, interpretation of the configuration for topic 86 is difficult.

It is possible to summarize results from these four topics. For topic 100, query expansion reduces the variation due to restatement of the topic as one would hope. For topic 78, query expansion also reduces the variation due to restatement but the two expansion systems do this differently. For topic 59, query expansion does not recognize one equivalence in the query statements, the equivalence between "storm-related" and "weather-related." For topic 86, query expansion fails in a more complex way.

4. CONCLUSIONS

The claim in this paper is that beyond differentiation of relevant and irrelevant documents, more insight can be obtained from the document identifiers that are part of the TREC system responses. In particular, we consider the return order of relevant documents compared by means of Spearman's coefficient of rank correlation. We have supported our claim by showing that for specific topics, the return order of relevant documents can help us understand the difference between systems with and without query expansion. This paper opens the door to many more possibilities for insights.

Our claim is not that a dissimilarity matrix computed from the return order of relevant documents is a substitute for a performance measure of the precision and recall variety. One important way of going beyond the analysis in this paper is extension to an analysis of both the dissimilarities in this paper and a selected performance measure such as average precision. Computing a new dissimilarity measure from the two is a possibility but perhaps not the best idea. Rather, one should realize that there may be topics for which the performance measure does little to distinguish the system returns but the return order of relevant documents is much more informative. There may be topics for which the converse is true. These are the topics for which further insight can be obtained by considering in addition to the usual performance measures, the return order of relevant documents.

One possibility would be to compute the performance difference between expansion and non-expansion systems for all the topics and use this series of numbers to pick topics to be looked at in terms of the return order of relevant documents. Such an approach seems necessary because looking individually at all 50 topics seems overwhelming in light of the four or five figures that each topic requires for interpretation. Thus, the return order of relevant documents would be the basis for analysis of the failures of query expansion.

One would like to summarize what is shown by our return order of relevant documents over all 50 topics. This is not as easy as with a performance measure that can be averaged over the topics. One can however, think of quantifying what is observed in the topics discussed above. In the case of Topic 100 in particular, one can think of a measure of scatter that would one could use to evaluate the effectiveness of query expansion. One could then compute such a measure for each topic and use it to summarize over topics. One could then rank topics by the effectiveness of query expansion and investigate a sampling of topics in detail. Such an investigation could be the next step.

REFERENCES

D. Banks, P. Over, and N. Zhang (1999). "Blind Men and Elephants: Six Approaches to TREC Data," *Information Retrieval* 1, 7-34.

M. W. Berry and M. Browne (1999). *Understanding Search Engines: Mathematical Modeling and Text Retrieval*, Philadelphia, PA: Society for Industrial and Applied Mathematics.

C. Buckley and J. Walz (2000). "The TREC-8 Query Track," In E. M. Voorhees and D. K. Harman, editors, *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*," pages 65-75, NIST Special Publication 500-246, Washington, DC: US Government Printing Office.

T. F. Cox and M. A. A. Cox (1994). *Multidimensional Scaling*, London: Chapman & Hall.

J. D. Gibbons (1985). *Nonparametric Statistical Inference*, New York: Marcel Dekker.

J. B. Kruskal and M. Wish (1978). *Multidimensional Scaling*, Newbury Park, CA: SAGE Publications.

M. Rorvig (1999). "Images of Similarity: A Visual Exploration of Optimal Similarity Metrics and Scaling Properties of TREC Topic-Document Sets," *Journal of the American Society for Information Science*, 50, 639-651.

W. N. Venables and B. D. Ripley (1999). *Modern Applied Statistics with S-PLUS, Third Edition*, New York: Springer-Verlag.

The TREC-9 Query Track

Chris Buckley
Sabir Research, Inc
chrisb@sabir.com

1 Introduction

The Query Track in TREC-9 is unlike the other tracks in TREC. The other tracks attempt to compare systems to determine the best approaches to solve the particular track problem. This comparison is normally done over a given set of topics, with a single query per topic. The Query Track, on the other hand, compares multiple queries on a single topic to determine which queries perform best with which systems. There is no emphasis on system-system comparisons: none of the participating systems were even the most advanced system from that particular participating group. Instead, the goal is to try and understand how the statement (query) of the user's information need (topic) affects retrieval.

Information Retrieval is a somewhat odd discipline. It's one where a human can do much better than any IR system, given infinite time and patience. Given any particular information need and some representative relevant documents, a user can often find an automatic retrieval strategy that does much better than an IR system. But, as any experienced IR system designer knows, implementing such a strategy may improve performance on this query and/or topic, but end up hurting performance on other types of queries and/or topics. Humans are remarkably adept at finding different ways to express similar ideas in both queries and documents; this variability is the heart of the difficulty of the information retrieval task.

The Query Track is an attempt to isolate some of the issues dealing with query and topic variability. Automatic IR systems perform tremendously differently across a typical IR task such as in the Web Track, but much of this variability is concealed by the evaluation averages. What is often quite surprising, especially to people just starting to look at IR, is the large variability in system performance across topics as compared to other systems. In a typical TREC task, no system is the best for all the topics in the task. It is extremely rare for any system to be above average for all the topics. Instead, the best system is normally above average for most of the topics, and best for maybe 5%-10% of the topics. It very often happens that quite below-average systems are also best for 5%-10% of the topics, but do poorly on the other topics. The Average Precision Histograms presented on the TREC evaluation result pages are an attempt to show what is happening at the individual topic level.

One of the major purposes of the Query Track is to try to understand how much of the system variability is due to issues of *how* the user's information need is being expressed (the query syntax), and how much is due to *what* the information need is (topic semantics).

1.1 Query vs Topic

For the purposes of this track, a *topic* is considered an information need of a user. It includes a full statement of what information is wanted as well as information the user knows that pertains to the request. A *query* is what the user actually types to a retrieval system. It is much shorter than a topic, but is the only direct information from the user that the system has. Topic 51 (the first topic used in the Query Track) is given below. A query corresponding to Topic 51 might be something as simple as "Airbus subsidies".

TOPIC 51

```
<top>
<head> Tipster Topic Description
<num> Number: 051
<dom> Domain: International Economics
<title> Topic: Airbus Subsidies
<desc> Description:Document will discuss government assistance to Airbus Industrie, or mention a trade dispute between Airbus and a
U.S. aircraft producer over the issue of subsidies.
<smry> Summary:Document will discuss government assistance to Airbus Industrie, or mention atrade dispute between Airbus and a
U.S. aircraft producer over the issue of subsidies.
<narr> Narrative:A relevant document will cite or discuss assistance to Airbus Industrie by the French, German, British or Spanish
government(s), or will discuss a trade dispute between Airbus or the European governments and a U.S. aircraft producer, most likely
Boeing Co. or McDonnell Douglas Corp., or the U.S.government, over federal subsidies to Airbus.
<con> Concept(s):
1. Airbus Industrie
2. European aircraft consortium, Messerschmitt-Boelkow-Blohm GmbH, British   Aerospace PLC, Aerospatiale, Construcciones
Aeronauticas S.A.
3. federal subsidies, government assistance, aid, loan, financing
4. trade dispute, trade controversy, trade tension
5. General Agreement on Tariffs and Trade (GATT) aircraft code
6. Trade Policy Review Group (TPRG)
7. complaint, objection
8. retaliation, anti-dumping duty petition, countervailing duty petition,   sanctions

 <def>  Definition(s): ...
```

1.2 Issues to Examine

There are a number of issues that we wish to examine in both last year's and this year's Query Track data, and in the future with the NIST Query Station. They include

- Can we distinguish between easy and hard queries/topics?
  - Are queries hard or are topics hard?
  - Even if we can distinguish this from the results, can NLP analysis of a query distinguish this before-hand?
- What categories of queries can potentially yield performance differences?
- Where do query performance differences come from?
  - Examine system vs topic vs query.

BEST COPY AVAILABLE

- Can we easily create test collections with large numbers of queries with judgments?

If we can answer these questions, then we may make it possible to improve retrieval systems dramatically.

2 Query Track Test Collection Creation

The construction of the Query Track test collection consists of 2 sub-tasks. In the first sub-task, groups take each of topics 51-100 from TREC 1 and create one or more queries based on the topic. In the second sub-task, each group runs one or more versions of their system on all the queries from all the groups. The results are then evaluated and analysis can begin!

2.1 Query Creation Sub-Task

Groups create one or more versions of each of TREC topics 51-100 in categories

- Very short: 2-4 words based on the topic and possibly a few relevant documents from TREC disk 2.
- Sentence: 1-2 sentences using topic and relevant documents.
- Sentence-Feedback only: 1-2 sentences using only the relevant documents. The aim is to increase vocabulary variability.

This is the second (and final) year of the Query Track. Last year there were five participating groups who produced 23 Query Sets. Each query set consisted of 50 queries corresponding to topics 51-100. Two of the Query Sets were not natural language (lists of weighted terms) and were not re-used. The other 21 Query Sets were used again this year. To this we added another 22 Query Sets, giving us a total of 43 Query Sets from 6 groups.

APL	INQ	Sab	Acs	Pir	Uof M
Johns Hopkins	Umass	Sabir	Acsys	Queens	Melbourne
Expert	Students	Expert	Expert	Expert	Expert
1 short 1 sent.	10 short 10 sent 10 fdbk	4 short 1 sent 1 fdbk	1 short	1 short	2 short 1 sent

Several versions of queries for topic 51 are given below. It was quite surprising how few duplicate queries there were. There were 2150 original queries. Of those, 1982 were unique after removing spaces, extra punctuation, and capitalization. After that, if hyphens were removed there were 1973 unique queries left. Every topic had at least 33 unique queries (out of the 43 possible.)

Sample of queries for Topic 51

•51 01 recent airbus issues
•51 02 Airbus subsidies dispute
•51 03 Airbus subsidy battle
•51 04 Airbus subsidies dispute
•51 05 U.S. Airbus subsidies
•51 06 What are the reactions of American companies to the trade dispute and how the dispute progresses?
•51 07 What are the issues being debated regarding complaints against Airbus Industrie?
•51 08 News related to the Airbus subsidy battle.
•51 09 U.S. and Europe dispute over Airbus subsidies
•51 10 Is European government risking trade conflicts over issue of Airbus subsidies?

2.2 Retrieval Sub-Task

After the Query Sets were constructed, they were distributed to all the groups to run one or more retrieval runs on the TREC Disk 1 document collection (about 510,000 documents). Six groups performed 18 retrieval runs:

- INQ: 3 runs
  - only query terms
  - query terms plus structure
  - query terms plus structure plus blind feedback
- SUN: 2 runs
  - Used two slightly different versions of their Question Answering Track engine
- Sab: 3 runs
  - query terms plus adjacency phrases
  - query terms plus phrases plus 7 terms expansion from blind feedback
  - query terms plus phrases plus 60 terms expansion
- UoM: 2 runs - no expansion
- hum: 7 runs
  - baseline, linguistic morphology
  - spelling correction (for words occurring in less than 10 documents)
  - no keywords in documents
  - varying idf weight (squared normally, but not here)
  - keep high frequency terms (normally dropped)
  - old version of software
- ok7: 1 run - no expansion, base run

The groups submitted the results (top 1000 documents retrieved for each query) to NIST for evaluation. There were a total of 774 runs: 18 system variants times 43 queries.

The runs were evaluated at NIST using trec_eval, concentrating on Mean Average Precision. The results of the initial evaluation were given to the six groups. This included

- Rankings of all documents (1.7 Gbytes in size)
- MAPs of all groups on all queries
- Various averages and standard deviations

These results are now publicly available at NIST on the TREC web site.

We can now compare systems on 2000 queries, making a qualitative difference in possible investigations. It has proven to be great tool for analyzing systems. Some of the differences among queries of a single topic pinpoint weaknesses in stemming, phrasing, hyphenation, and spelling correction. Other differences show that some systems are able to handle an entire topic better than other systems, while being worse on other topics. This comparison of differences due to syntax (queries) and semantics (topics) should prove very interesting.

The short-term goal of the Query Track has been to gather raw data for analysis. The long-term depends on you, the members of the community. You can both contribute more data, submitting runs of your system to the Query Station, and contribute your analysis.

Halfway To Question Answering

W. A. Woods Stephen Green Paul Martin
Ann Houston
Sun Microsystems Laboratories
1 Network Drive
Burlington, MA 01803
{William.Woods,Stephen.Green,Paul.Martin,Ann.Houston}@east.sun.com

1 Introduction

The conceptual indexing and retrieval system at Sun Microsystems Laboratories (Woods et al., 2000) is designed to help people find specific answers to specific questions in unrestricted text. It uses a combination of syntactic, semantic, and morphological knowledge, together with taxonomic subsumption techniques, to address differences in terminology between a user's queries and the material that may answer them. At indexing time, the system builds a conceptual taxonomy of all the words and phrases in the indexed material. This taxonomy is based on the morphological structure of words, the syntactic structure of phrases, and semantic relations between meanings of words that it knows in its lexicon. At query time, the system automatically retrieves any concepts that are subsumed by (i.e., are more specific than) the user's query terms, according to this taxonomy.

The system uses a penalty-based relaxation-ranking method to locate and rank potentially relevant passages in the material. It provides a user with feedback on why passages were retrieved, so that irrelevant passages can be quickly skipped over, and it provides information about where query terms fit in its conceptual taxonomy, so that users can see opportunities for generalizing queries if the first request is not successful or more information is desired. This methodology, which falls somewhere between traditional document retrieval and TREC-style question answering, has proven to be very effective in reducing the time required for people to find information in online material (Woods et al., 2000).

For the most part, the conceptual indexing project has focussed on finding techniques for improving human search productivity, rather than dealing with the problems of large collections. However, one experimental version of this system, which we internally call Nova, has recently reached the point where it can index collections the size of the TREC corpora. In addition to Nova, we have a pilot system that includes much stronger morphology and phrase extraction components and a newer lexicon; however, the pilot system is currently limited in the amount of text that it can process.

We decided to enter a system based on Nova in the question-answering track of this year's TREC competition in order to see how far we would have to go to transform Nova into a full question-answering system. From our experience with Nova, it seemed that it might provide a good first-stage retrieval for a question-answering system because it is able to find specific passages where the terms of a request occur near each other and can automatically deal with some of the paraphrase differences between the wording of a request and the wording of a relevant passage. Nova is able to find good passages directly from the conceptual index, without first performing a separate document retrieval step.

BEST COPY AVAILABLE

We entered the TREC competition very late in the game, so we had only a couple of weeks to download and index the TREC collections, implement some quick extensions to Nova, and run an experiment. We indexed the TREC material using an option in Nova that indexes all word occurrences and their subsumers. We had no previous exposure to any of the TREC collections, so Nova's lexicon and conceptual taxonomy have had no previous adaptation to this material.

We discovered that with the addition of a simple automatic query formulation algorithm, Nova performs modestly well at the 250-byte question answering task, finding a correct answer in the top 5 hits roughly half of the time. The remaining cases tend to fall into three categories:

1. Cases where a key passage was not found due to a paraphrase relationship that was not yet covered in our conceptual taxonomy. For example, "The first Russian astronaut to spacewalk" was described as a "Soviet cosmonaut" in a desired passage, and our taxonomy doesn't yet know that Soviet means Russian or that cosmonaut is a kind of astronaut. If these facts were in our lexicon, then Nova would have gotten the correct answer at rank 1.

2. Cases where glitches in the current version of Nova or our experimental lashup (which combined Nova and parts of the pilot system) caused it to fail to find things that it should have. See the failure analysis in section 5.2 below for examples.

3. Cases where nonessential terms in the query could not be found in the desired passages, but could be found elsewhere in the material in combination with almost all of the other elements of the query, but missing an essential element. For example, in question 411, "What tourist attractions are there in Reims?", we found numerous tourist attractions elsewhere, but none of the passages that mention Reims contains either *tourist* or *attraction* or anything subsumed by them in our taxonomy.

We also discovered that the Nova system itself is a highly useful tool for investigating why a given passage was not retrieved. With a Nova index to the entire collection, we can quickly find passages containing combinations of specified terms, and can use general terms in the request that may subsume more specific terms in the text. We can view these passages in decreasing order of quality, and jump quickly to the corresponding positions in the source documents. We can also quickly survey how given terms are used in the collection and how they relate to other terms in our taxonomy. For example, it took only a few tries to discover the "Soviet cosmonaut" variation of the above-mentioned request.

2 Getting from Nova to Question Answering

As mentioned previously, Nova does something between document retrieval and question answering: It identifies useful places in the documents for a human searcher to look, and it provides information to help a person make quick assessments about whether a passage is likely to be relevant, but it doesn't really understand either the question or the answer. The TREC QA task requires more than that.

Nova is missing two key components necessary to perform the TREC question-answering task:

1. A query-formulation component to determine the answer type of the desired result and to transform the natural language question into a more effective query.

2. An answer-location component for locating answers in or near the retrieved passages.

For this competition, we provided Nova with simple baseline versions of these two components.

The query-formulation component is based on our pilot system. The query formulator interprets the question words and the format of the question to determine the desired answer type. It also either replaces the question word in the query with the desired answer type or simply removes it from the request.

The answer-location component determines a 250-byte passage of text that may contain the answer to the question. Nova returns passages that typically range from a single term to a passage the size of a paragraph or more and also returns approximately 200 bytes of context on either side of the passage. We have observed that the answer to a Nova query is often either in the retrieved passage or the neighboring context. The task of the answer-location component is to select one or two 250-byte passages from the (typically 500 to 1000 byte) passage returned by Nova.

This approach differs from most systems in the QA track in that we used Nova to index and search the entire TREC 9 collection, eliminating the stage of using a preliminary document retrieval system to select a subset of the collection for further processing.

3 Run Details

3.1 Query formulation

Questions have a logical structure consisting of an answer type and a relational condition. Answering a question consists in finding an instance of the answer type that satisfies the relational condition. Like other systems, we implemented a function to interpret a question into such a structure. For example, for one of the TREC 8 questions, "What was the monetary value of the Nobel Peace Prize in 1989?", this function produced the structure in figure 1. This structure indicates that a number is sought as the answer and that `MONETARY VALUE OF NOBEL PEACE PRIZE IN 1989` is to be used as the Nova query.

```
(NUMBER (MONETARY VALUE OF NOBEL PEACE PRIZE IN 1989))
```

Figure 1: Question structure produced by query formulation.

The query-formulation stage determines the answer type, replaces the question word with the answer type (or sometimes just removes it), and often changes the wording of the relational condition in one or more of the following ways:

- Shorten it if necessary, by dropping less important terms. This step is necessary because Nova's design point has been short questions and the current implementation does not use more than ten words of a request. Previously, the longest query we had encountered was eight words, and most requests consisted of three or fewer words.
- Generalize some terms (e.g., *high* for *tall*), substitute base forms for inflected forms (e.g., *principle* for *principles*), and drop some "noise" terms.

Like other TREC QA systems, for example, (Moldovan et al., 2000; Breck et al., 2000), this first stage consists of a dispatch table based on the question words and their pattern of use in the question. The patterns in Table 1 roughly characterize the rules for determining the answer type.

Question word	Answer type
whose	person
(who whom)	person
(which what)	depends on subsequent words
prep (which what whom)	do case analysis
when	date
where	location
why	reason
how (many much long)	number
how *adjective*	number
how	way
name	name
otherwise	look for embedded question words

Table 1: Dispatch table for question words.

These rules are similar to those used in other TREC systems. In addition, we performed a number of other transformations on the rest of the question:

- Combine elements like *U.S* from *U . S* and strip out other punctuation.
- Delete elements like *name of, most of, day of week, time of day, can be, may be, can not, must not, have been, take place, held at, does it take*.
- Replace inflected forms of words with their base forms.
- Generalize certain words (e.g., *tall* to *high*).
- If query is too long, skip parenthetical comments in parentheses, unless they are names.
- If query is still too long, delete words from a stop word list.
- If query is still too long, delete less important elements from a priority-ordered list until query is short enough or it runs out of things to delete (in which case simply truncate it to 10 words).

These rules were put together quickly based on examining some of the TREC 8 questions.

3.2 Answer location

For the answer-location component, we tried two simple approaches, embodied in the runs we labeled SunOne and SunToo, both of which used the query-formulation component described above. For SunOne, we used the pilot system to locate all occurrences of the desired answer type that occurred in Nova's retrieved passage or its neighboring context and then selected one or two 250-byte passages to include as many candidate answers as possible. No effort was made to determine the actual relationship between these candidates and the content of the query. For SunToo we simply truncated passages longer than 250 bytes to the leftmost 250 bytes, and symmetrically extended shorter passages to include a full 250 bytes.

We thought these approaches might give a useful approximate answer location because we had noticed that Nova often gets the desired information as the first hit[1], either in the passage itself or in the neighboring context region that Nova returns with the passage.

There are a number of different situations in which one or more instances of the desired answer type can occur (or not occur) in or near the retrieved passages, so the following rules were used in the SunOne case to select a 250 byte passage in the various cases:

1. No answer type in or near passage and passage is not longer than 250 bytes. Extend both ends of the passage.
2. No answer type in or near passage and passage is longer than 250 bytes. Shrink the passage by cutting off the right end.
3. Answer types at both ends but all found answer types fit within 250 bytes. Extend both ends of the passage to include all answer types.
4. Found answer types span more than 250 bytes, so split the passage into two pieces (the left and right halves) and slightly favor one (usually the left) in the ranking.
5. Answer types at left end only. Shift passage to the left.
6. Answer types at right end only. Shift passage to the right.

These rules indicate how to move the ends of the passages that Nova returns in order to determine the 250 bytes to be used. In case number 4, instances of the answer type were found sufficiently far apart that the Nova passage was split into two and both were generated as candidates, with the leftmost being favored in most cases, and the rightmost in some cases.

3.3 Results

The results of the two runs are shown in Table 2, where the scores are labeled to indicate the strict and lenient human judgements and the scores that were derived for the training set from the automatic eval script provided for TREC 8 as extended by us to include some answers we found in the TREC 9 material that were not covered in the script. Interestingly, although the SunOne strategy outperformed the SunToo strategy in our testing on the TREC 8 questions, they perform almost equally on the TREC 9 test, with the simple SunToo case slightly outperforming SunOne. For both strategies, the TREC 9 questions were substantially more difficult than the TREC 8 questions.

After the formal submission, we discovered a low-level paging bug in the Nova retrieval system, the effect of which was to make a portion of Nova's conceptual taxonomy invisible to the search process. This resulted in noticeable errors in the system performance such as sometimes failing to find a hit on an inflected form of a word when the query term was a base form of that word, a capability that Nova generally does quite well. After the formal submission and after receiving the eval script that allows us to automatically score a run, we reran the system with this bug fixed but with no other changes. We rescored both the submitted run and the run with the bug fix, using the eval script, to determine the effect of the bug and to calibrate the difference between the eval script scores and the human judgement scores.

---

[1] Getting the correct information near the top and making it easy for a person to skip over irrelevant passages has been the main goal; getting it to actually be first was considered a bonus.

The results are shown in Table 3. In this table, we estimated the equivalent strict human judgements for the system with the bug fixed by assuming that the ratio of the eval script MRR scores to the human judgement MRR scores is a constant and similarly for the success rate scores. Notice that although the result is a 10% improvement in MRR and 12% improvement in success rate, this improvement is the net result of improving on some questions and losing the answers (from the top 5) or moving down in the ranking for some others. Because the nature of the bug was to hide a portion of the taxonomy, this turns out to be an inadvertent experiment that reconfirms the results of several previous experiments which show that increasing the amount of knowledge in the taxonomy results in an improvement in the retrieval results for this technology.

	Mean reciprocal rank	Success rate at 5 hits
training set:	TREC 8 questions	(on TREC 9 data)
SunOne	0.50 (eval script)	64.6% (eval script) (71.2% at 10 hits)
SunToo	0.44 (eval script)	55.1% (eval script)
submitted:	TREC 9 questions	(on TREC 9 data)
SunOne	0.340 (strict) 0.348 (lenient)	46.2% (strict) 47.4% (lenient)
SunToo	0.345 (strict) 0.354 (lenient)	46.9% (strict) 47.5% (lenient)

Table 2: Results of two Sun runs on TREC 8 and TREC 9 questions.

	Mean reciprocal rank	Success rate at 5 hits
submitted:	TREC 9 questions	(with paging bug)
SunToo	0.345 (strict) 0.369 (eval script)	46.9% (strict) 48.39% (eval script)
corrected:	TREC 9 questions	(with bug fixed)
SunToo	0.380 (strict*) 0.406 (eval script)	52.6% (strict*) 54.25% (eval script)
improvement:	10%	12%
net gain:	picked up 73 new answers	and lost 33 old ones; net gain 40

Table 3: Results of correcting the paging bug. (* indicates estimated strict results)

4 Question Analysis

Because our results depend in part on our ability to formulate good queries from the questions, and this may depend on the type of the question, it is informative to look at the distribution of question types in the TREC queries and our performance on the different types. Table 4 shows the number (and percentage) of different question types from both TREC 8 and TREC 9 and our corresponding performance.

Note that two of the three question types for which our success rates were the lowest (*how* and *what*) are those which have the greatest diversity of entries in Table 1, and are cases where our analysis to identify the answer type for the question is somewhat limited. We also note that there were a variety of new kinds of *what* questions in TREC 9 that did not occur in TREC 8 and more than half of the TREC 9 questions are *what* questions. This partly explains why TREC 9 is a substantially more difficult task.

Question type	TREC 8		TREC 9		TREC 9 results				
how	31	(15.9%)	54	(8.4%)	SunOne:	MRR:	0.122	SR:	25.9%
					SunToo:	MRR:	0.162	SR:	29.6%
what	65	(33.3%)	374	(58.3%)	SunOne:	MRR:	0.315	SR:	43.3%
					SunToo:	MRR:	0.322	SR:	43.3%
when	19	(9.7%)	49	(7.6%)	SunOne:	MRR:	0.290	SR:	42.9%
					SunToo:	MRR:	0.319	SR:	46.9%
where	21	(10.8%)	72	(11.2%)	SunOne:	MRR:	0.398	SR:	54.2%
					SunToo:	MRR:	0.447	SR:	59.7%
which	10	(5.1%)	13	(2.0%)	SunOne:	MRR:	0.359	SR:	46.2%
					SunToo:	MRR:	0.308	SR:	46.2%
who	47	(24.1%)	117	(18.3%)	SunOne:	MRR:	0.500	SR:	60.7%
					SunToo:	MRR:	0.450	SR:	58.1%
why	2	(1.0%)	3	(0.5%)	SunOne:	MRR:	0.444	SR:	66.7%
					SunToo:	MRR:	0.500	SR:	66.7%
other	5	(2.5%)	0	(0.0%)					

Table 4: Distribution of question types for TREC 8 and TREC 9.

5 Results Analysis

In order to try to understand the results of this experiment, we have begun to look in some detail at the behavior of the experimental lashup. So far we have done three analyses: a random sample study of 10 randomly chosen questions, a failure analysis of a selected set of questions on which we did less well than other systems, and a comparison of the results of different paraphrases.

For the random sample, it turns out that all of the cases where no correct answer was found are cases where Nova failed to find at least one of the requested terms in the best ranked passage. This is not true in general, but it is a strongly correlated indicator. Table 5 shows the mean reciprocal rank and the success rates broken down by whether the first hit has no missing terms or the first hit has at least one missing term. This data suggests that one way to use Nova for question answering would be to reformulate the question and ask again when there is a missing term in the best hit. This is typical of the way that humans use Nova.

	Number of questions	Mean reciprocal rank	Success rate
SunOne: N[illegible]ssing term	444	0.417 (strict)	55.9%
SunOne: Missing term	238	0.196 (strict)	28.2%
SunToo: No missing term	444	0.435 (strict)	57.2%
SunToo: Missing term	238	0.177 (strict)	27.7%

Table 5: Performance with respect to missing terms in best hit.

Like other systems in TREC 8, we have observed a bimodal distribution of results — we tend to either do fairly well on a question or we miss it entirely.

5.1 Random sample analysis

Table 6 shows the results for 10 randomly chosen questions from the TREC 9 test set (using the strict SunOne results, and showing the answer type and Nova query that were used).

Num	Question	Nova query	Result
402	What nationality was Jackson Pollock?	(NATIONALITY (NATIONALITY JACKSON POLLOCK))	Rank 1
455	What is Colin Powell best known for?	(THING (COLIN POWELL BEST KNOW FOR))	Rank 5
459	When was John D. Rockefeller born?	(DATE (JOHN D ROCKEFELLER BORN))	Rank 2
487	What was the name of the movie that starred Sharon Stone and Arnold Schwarzenegger?	(THING (MOVIE STAR SHARON STONE ARNOLD SCHWARZENEGGER))	Missing "Sharon" in first hit
576	What is the name of Joan Jett's band?	(THING (JOAN JETT BAND))	Rank 3
616	What is the purpose of a car bra?	(THING (PURPOSE OF CAR BRA))	Missing "bra" in first hit
657	In what country is a stuck-out tongue a friendly greeting?	(COUNTRY (IN COUNTRY STUCK-OUT TONGUE FRIENDLY GREET))	Missing "stuck-out" and "tongue" in first hit
711	What are the names of the tourist attractions in Reims?	(THING (NAMES OF TOURIST ATTRACTION IN REIMS))	Missing "Reims" in first hit
803	What king signed the Magna Carta?	(THING (KING SIGN MAGNA CARTA))	Rank 1
865	What is the meaning of caliente (in English)?	(THING (MEAN OF CALIENTE IN ENGLISH))	Missing "caliente" in first hit

Table 6: Random sample of questions and results.

The questions in this sample for which a correct answer was not found, all of which (as mentioned) missed at least one term in the best passage Nova could find, were investigated using the Nova system interactively to try to find passages that would answer the question. The results are as follows:

For question 487, "What was the name of the movie that starred Sharon Stone and Arnold Schwarzenegger?," a correct passage is found at rank 1 if "movie" is replaced by "film", even though it is missing a term for "star". Our pilot system taxonomy knows that "movie" is equivalent to a sense of "film", but the taxonomy used by Nova doesn't know this yet.

For question 616, "What is the purpose of a car bra?," the only thing we could find was a "stealth car bra" which fools police radar (rank 3 for "car bra" and repeated multiple times in the collection). There were no other occurrences of car bra in the collection, except for one consumer complaint about having purchased one. Apparently the human judges considered the "stealth car bra" correct, but this is not the purpose of an ordinary car bra.

For question 657, "In what country is a stuck-out tongue a friendly greeting?," the only instances of stuck-out tongue were for Mr. Yuk, a children's symbol for poison. No entrant got a correct answer for this question.

For question 711, "What are the names of the tourist attractions in Reims?," the submitted query included the "names of", which favored hits that mentioned "names" at the expense of "Reims". Dropping that and simply asking "tourist attraction in Reims" still was missing Reims. Trying "sight in Reims" didn't do it either. Simply asking for Reims gets Reims Cathedral in the third hit.

For question 865, "What is the meaning of caliente (in English)?," the query "caliente" gets a correct answer at rank 10.

5.2 Failure analysis

We looked in some detail at cases where other systems seem to have done better than Nova. One strong pattern that emerged is that a number of glitches of various sorts blocked many passages from being retrieved that would otherwise have been handled by this methodology. Many questions failed due to bugs and/or missing facts in our lexicon, and the fact that we used a newer version of the lexicon and morphology for the query-formulation and answer-location parts of the system than was used in the version of Nova that we were using to find the passages.

For example, in question 515, the word *Sinemet*, which was not in our lexicon, was aggressively analyzed morphologically (Woods, 2000) as the past tense of *sinemeet* and the query-formulation component substituted *sinemeet* for *sinemet* in the query given to Nova. Although wrong, this would not have caused a problem if the same version of the morphology had been used in Nova, since *Sinemeet* would have subsumed *Sinemet* and retrieved the passage correctly. (The morphological component has since been fixed to avoid such analyses of words ending in *met*.) With this fix, the correct passage is found at rank 2.

Another pattern that emerged is that Nova is currently very generous in its subsumption analysis, allowing a subsumption if any sense of a word is subsumed, without any attempt to disambiguate the sense of the word in the text. Since Nova is also knowledgeable about many names that are also ordinary words in English (like *bill* and *woods* and *green*), many of which are also city names, it finds a number of subsumptions of ordinary words under *person* and *place*. This sometimes interacts badly with the question-answering strategy.

Finally, as described above, in the version of Nova that we ran, there was a low-level paging bug that effectively hid part of the conceptual taxonomy and, among other things, failed to retrieve inflected forms of some words in some cases (e.g., *hexagons* was not retrieved in response to a query including *hexagon*). We have identified a number of queries where this bug kept us from finding answers, and the experiment described in Table 3 shows that fixing this single bug results in a 10-12% improvement.

5.3 Sensitivity to paraphrase variations

To assess the sensitivity of this methodology to paraphrase variations, we compared the results for different questions in the TREC 9 paraphrase sets. Table 7 gives an overview of how well the system handled different paraphrases on a question-by-question basis. This table shows the question numbers of questions in the TREC 9 paraphrase sets, followed by the respective rank scores at which SunToo found answers (the NILs are for questions for which there was no result reported in the TREC 9 results). From these numbers you can see that SunToo did consistently well or consistently poorly on many paraphrases, e.g.:

(450 863 864 865 866) (0 0 0 0 0)
(454 830 831 832 833 834) (1 1 1 1 1 1)

However, there are other cases in which the details of the paraphrase made a big difference, e.g.:
(394 808 809 810) (1 0 0 1)
394 What is the longest word in the English language?
808 What English word has the most letters?
809 What English word contains the most letters?
810 What is the longest English word?

Paraphrase set	Reciprocal ranks	Paraphrase set	Reciprocal ranks
(201 732 733 734)	(0 0 0 0)	(203 728 729 730 731)	(0 0 0 1/4 1/2)
(393 805 806 807)	(1 1 0 0)	(394 808 809 810)	(1 0 0 1)
(396 811 812 813)	(0 NIL 0 0)	(397 814)	(0 0)
(398 815 816 817)	(0 0 0 0)	(400 867 868 869)	(1 0 0 0)
(402 870 871 872)	(1 1 1 1)	(403 873 874 875 876 877)	(0 0 0 1 0 0)
(404 878 879 880 881 882)	(0 0 0 0 0 0)	(405 883 884 885 886 887)	(1/4 0 1 1 0 1)
(406 888 889 890 891)	(0 1/2 1/3 1/4 1/2)	(407 892 893)	(1/5 1 1/2)
(408 701 702 703 704)	(1/2 1/2 0 1 1/2)	(409 705 706 707 708)	(1 1/2 0 0 1/2)
(410 709 710)	(0 0 0)	(411 711 712 713 714 715 716 717)	(0 0 0 0 0 0 0 0)
(412 718 719 720 721 722)	(0 0 0 0 0 0)	(413 723 724 725 726 727)	(1 1 1/2 1/5 0 1/2)
(415 735 736)	(0 1 1/2)	(416 737 738 739 740 741)	(0 0 0 0 0 1)
(417 742 743)	(1 1 1)	(418 744 745)	(1 1 1)
(419 746 747 748 749)	(1 1 0 1 1)	(421 750 751 752)	(1 0 0 0)
(423 753 754 755 756 757)	(0 0 0 0 0 0)	(424 758 759 760 761)	(0 0 0 0 0)
(425 762 763 764 765)	(1/2 0 1 0 1/2)	(426 766 767 768)	(1/2 0 1/3 1/2)
(427 769 770 771)	(0 0 0 0)	(428 772 773 774 775 776 777)	(1 1 1/3 1 1 1 1)
(429 778 779 780)	(1/3 1/3 1/3 1)	(431 781 782 783 784)	(0 0 0 0 1)
(433 785 786)	(0 0 0)	(435 787 788 789)	(1/2 0 0 0)
(436 790 791 792 793)	(0 1/3 0 1/3 0)	(437 794 795 796)	(0 NIL 0 NIL)
(440 797 798 799)	(1 0 0 0)	(441 800 801 802 803 804)	(1/2 1 1 1/2 1 1)
(442 844 845 846)	(1 0 1/5 1)	(444 847 848 849 850)	(1/5 1 1 1 1/2)
(445 851 852 853)	(0 1/5 0 1)	(446 854 855)	(1/5 1/2 1/2)
(448 856 857 858 859)	(0 1 0 0 1)	(449 860 861 862)	(1 0 1 1)
(450 863 864 865 866)	(0 0 0 0 0)	(451 818 819 820 821 822)	(1 0 0 0 0 0)
(452 823 824 825 826 827)	(1 0 1 1 1 1)	(453 828 829)	(1/3 1 1)
(454 830 831 832 833 834)	(1 1 1 1 1 1)	(455 835 836 837 838)	(0 1/4 0 1 1/2)
(456 839 840 841 842)	(1 0 0 1/2 1)	(458 843)	(0 1)

Table 7: Paraphrase sets and respective results.

A detailed analysis of these cases reveals the kinds of paraphrase variations that the system does not yet handle. In the above example, the system doesn't have a way to relate "the longest word" to "the word containing the most letters".

6 Query Track Experiment

Although we did not intend to participate in the TREC Query Track, and Nova is not a document retrieval system, we responded to the NIST request to run our system on the query track queries. We did this by modifying Nova to simply return the document id without the passage-specific information and to give each document the score and rank of its best passage. Previous experiments comparing this kind of modified passage retrieval to more traditional document retrieval systems had shown that the passage-retrieval-based system found relevant documents that the traditional system did not, and vice-versa[2]. The former tends to find documents that contain on-point passages even when the document overall may be about something else, while the latter tends to find documents that make repeated references to the terms in the query. For example, in the man-page experiment in (Woods et al., 2000), the system found the File Manager man page as an answer to "print a file" because of a paragraph explaining the function of the "print" button in the file manager, but the human judge that had previously determined the set of relevant documents hadn't thought of the File Manager man page as a relevant document.

We tried this modified Nova system in two versions: one (SUN) that took the query as given and passed it to Nova, and another (Sunl) that first applied the query-formulation procedure described in section 3.1 and passed the result to Nova. The latter performed better on this task because the former effectively truncates queries beyond the first 10 words and many of the queries have more than 10 words.

Since Nova was designed for specific information requests that need only one answer, the goal has been to get the best relevant passage in the first ten hits. Multiple hits for the same question have not been important. Consequently, we sacrifice high recall for the ability to drill in on precise content, and we don't score well on the MAP measure because of its dependence on recall. The figure of merit that interests us is the success rate, and that is what we measured in this experiment.

	Success rate at					
	20 docs		10 docs		5 docs	
Average for	Sunl	SUN	Sunl	SUN	Sunl	SUN
All Sets	74.56%	66.00%	64.70%	56.65%	54.37%	46.65%
Verbose Sets	70.83%	56.00%	60.50%	46.33%	50.33%	36.50%
Concise Sets	79.26%	78.63%	70.00%	69.68%	59.47%	59.47%

Table 8: Success rates for verbose and concise query sets.

The most salient difference between the query sets is the difference between the sentence cases, which have relatively long questions, complete with question words (e.g., Question 94 in set INQ2a "What kind of illegal activities can be performed with the aid of computers"), and the rest, which have relatively short topic descriptions (e.g., Question 94 in set INQ1j "Illegal computer activity"). In general, the Nova system works better on the concise topics, and more specifically on concise topics that relate to specific information. Table 8 gives the success rates for the two Query Track runs and the breakdown for verbose and concise query sets.

---

[2] Given this fact, it should be noted that because the relevance judgments for this experiment are based on previous TREC results, many of the "non-relevant" documents that we retrieved may actually be relevant, but were not found in the previous TREC results. Indeed, we have found a number of cases where we retrieved documents that were not judged relevant but appeared relevant to us.

7 Conclusions

Although the Nova system is not a question answering system, it is an interesting alternative for the first-phase retrieval that precedes more detailed linguistic processing in most question answering approaches. An experimental lashup using Nova with a simple query-formulation algorithm gets the correct answer in the top 5 choices approximately half the time on the TREC 9 task, without any real understanding of either the questions or the passages that it retrieves and using general-purpose linguistic knowledge that has had no previous exposure to the TREC material. The system could be improved greatly by expanding its knowledge and eliminating bugs in the experimental lashup. However, to achieve a real question-answering capability, we clearly need to move beyond the simple baseline strategies used here for query formulation and answer location.

In addition, the Nova system is a useful tool for analyzing the results of these experiments by helping a researcher find passages that could answer a question, so that the reasons for failure can be understood.

References

(Breck et al., 2000) Eric Breck, John Burger, Lisa Ferro, David House, Marc Light, and Inderjeet Mani. A Sys Called Qanda. In *Proceedings of The Eighth Text REtrieval Conference (TREC 8)*. National Institute of Standards and Technology, 2000.

(Moldovan et al., 2000) Dan Moldovan, Sanda Harabagiu, Marius Paşca, Rada Mihalcea, Richard Goodrum, Roxana Girju, and Vasile Rus. LASSO: A Tool for Surfing the Answer Net. In *Proceedings of The Eighth Text REtrieval Conference (TREC 8)*. National Institute of Standards and Technology, 2000.

(Singhal et al., 2000) Amit Singhal, Steve Abney, Michiel Bacchiani, Michael Collins, Don Hindle, and Fernando Pereira. AT&T at TREC-8. In *Proceedings of The Eighth Text REtrieval Conference (TREC 8)*. National Institute of Standards and Technology, 2000.

(Srihari and Li, 2000) Rohini Srihari and Wei Li. Information Extraction Supported Question Answering. In *Proceedings of The Eighth Text REtrieval Conference (TREC 8)*. National Institute of Standards and Technology, 2000.

(Voorhees and Harman, 2000) E.M. Voorhees and D.K. Harman, editors. *Proceedings of The Eighth Text REtrieval Conference (TREC 8)*. National Institute of Standards and Technology, 2000.

(Woods et al., 2000) William A. Woods, Lawrence A. Bookman, Ann Houston, Robert J. Kuhns, Paul Martin, and Stephen Green. Linguistic Knowledge can Improve Information Retrieval. In *Sixth Annual Applied Natural Language Processing Conference*, pages 262–267. Association for Computational Linguistics, 2000.

(Woods, 2000) William A. Woods. Aggressive Morphology for Robust Lexical Coverage. In *Sixth Annual Applied Natural Language Processing Conference*, pages 218–223. Association for Computational Linguistics, 2000.

Question Answering :
CNLP at the TREC-9 Question Answering Track

Anne Diekema, Xiaoyong Liu, Jiangping Chen, Hudong Wang, Nancy McCracken,
Ozgur Yilmazel, and Elizabeth D. Liddy

Center for Natural Language Processing
Syracuse University, School of Information Studies
4-206 Center for Science and Technology
Syracuse, NY 1324-4100
`{diekemar,xliu03,jchen06,hwang07,njm,oyilmaz,liddy}@syr.edu`

Abstract

This paper describes a question answering system that automatically finds answers to questions in a large collection of documents. The prototype CNLP question answering system was developed for participation in the TREC-9 question answering track. The system uses a two-stage retrieval approach to answer finding based on keyword and named entity matching. Results indicate that the system ranks correct answers high (mostly rank 1), provided that an answer to the question was found. Performance figures and further analyses are included.

1. Introduction

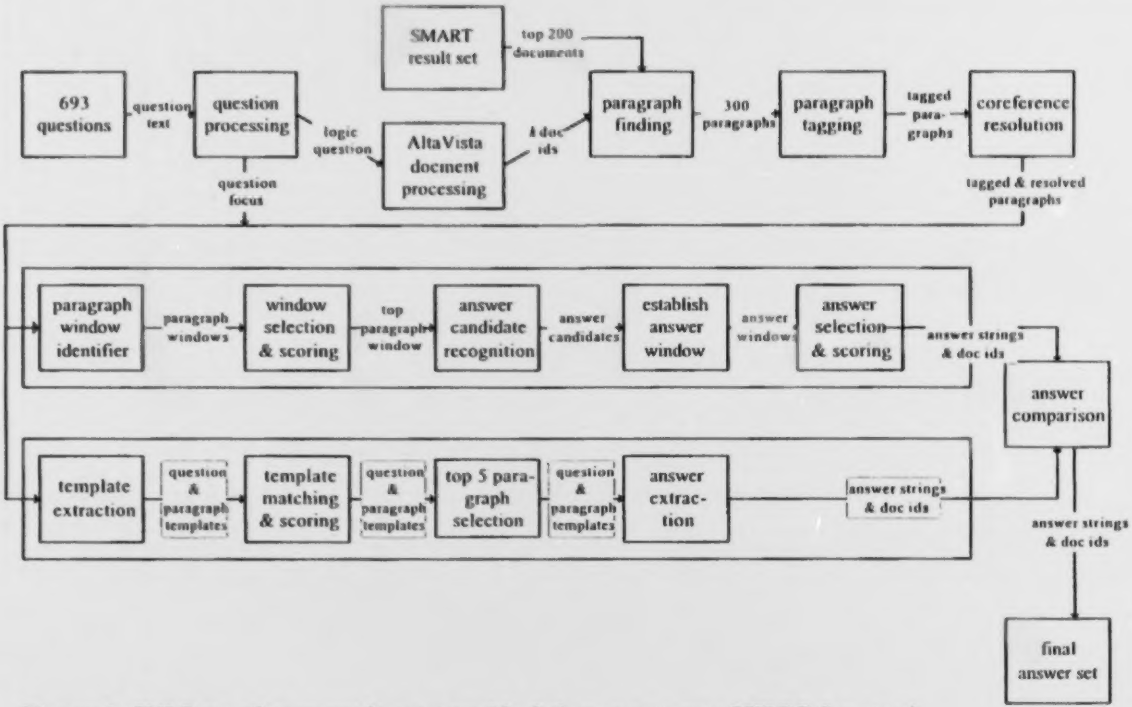
Question answering is not typically found in traditional information retrieval systems. In information retrieval, the system presents the user with a list of relevant documents in response to the query. The user then reviews these documents in search of the information that prompted the original search. It is not surprising therefore that, especially for short questions, people tend to ask their peers or forego the answer rather than expending time and effort with an information retrieval system. [3] Ideally, question answering helps users in their information finding task by providing exact answers rather than a ranked list of documents that may contain the answer.

The TREC question-answering track fosters question-answering research. Question-answering systems are not as well developed as information retrieval systems, especially for domain independent questions. As first-time participants, the Center for Natural Language Processing (CNLP) developed a question- answering system to deal with domain independent questions.

The CNLP question answering system uses a two-stage retrieval approach to answer-finding based on keyword, entity, and template matching (see figure 1). In answering a question, the system first creates a logical query representation of the question that is used for the initial information retrieval step. Additional modules take the retrieved documents for further processing and answer finding. Answer finding uses two different approaches after which answer triangulation takes place to select the most likely answer. The first approach to answer finding is based on keyword and entity matching and the second on template matching. Currently only the keyword and entity matching answer-finding approach has been implemented. A detailed system overview can be found in section 3.

2. Problem description

Participants in the question-answering track were provided with 693 questions that originated from search engine logs. The initial question set of 693 questions was reduced to 682 questions after 11 questions were discarded by the National Institute for Standards and Technology (NIST). The remaining questions were mostly fact-based and required short answers only (see figure 2). The base set of questions consisted of 500 questions. For 54 questions, slight variations were created resulting in an additional 193 questions. Answers to all 693 questions had to be retrieved automatically from approximately 3 gigabytes of data. Sources of the data were: AP newswire 1988-1990 (728 Mb), Wall Street Journal 1987-1992 (509 Mb), San Jose Mercury News 1991 (287 Mb), Financial Times 1991-1994 (564 Mb), Los Angeles Times 1989, 1990 (475 Mb), Foreign Broadcast Information Service 1996 (470 Mb).



SMART result set
top 200 documents
693 questions
question text
question processing
logic question
AltaVista docment processing
& doc ids
paragraph finding
300 paragraphs
paragraph tagging
tagged para-graphs
coreference resolution
question focus
tagged & resolved paragraphs
paragraph window identifier
paragraph windows
window selection & scoring
top paragraph window
answer candidate recognition
answer candidates
establish answer window
answer windows
answer selection & scoring
answer strings & doc ids
answer comparison
template extraction
question & paragraph templates
template matching & scoring
question & paragraph templates
top 5 para-graph selection
question & paragraph templates
answer extrac-tion
answer strings & doc ids
answer strings & doc ids
final answer set

Figure 1. CNLP question answering system (shaded areas not part of TREC-9 system).

For each question, up to five ranked answer submissions were permitted, with the system producing the most likely answer ranked first. The maximum length of the answer string for a retrieval run was either 50 bytes or 250 bytes. An response to a question consisted of the question number, the document ID of the document containing the answer, rank, run name, and the answer string itself. The submitted answer strings were evaluated by NIST' s human assessors for correctness. [6]

	TREC-9 question answering questions
Base question	419: Who was Jane Goodall?
Question variants	746: What is Jane Goodall famous for? 747: What is Jane Goodall known for? 748: Why is Jane Goodall famous? 749: What made Jane Goodall famous?
Answer string (50 bytes)	748 AP880225-0129 1 80.90 SUT9p2c3c050 for her 28 years of chimpanzee research

Figure 2. Examples of TREC-9 questions.

3. System overview

The prototype of the CNLP question-answering system consists of four different processes: question processing, document processing, paragraph finding, and keyword and entity based answer finding. Each of the processes is described in detail below.

3.1 Question processing

During question processing, the system converts the question into a logical query representation used for first stage information retrieval and the system determines the focus of each question used for answer finding. Question processing takes place in our Language-to-Logic or L2L module. The L2L process for the question-answering track is optimized for retrieval using the AltaVista search engine (see section 3.2), and includes a focus recognizer. For example, the question "What was the monetary value of the Nobel Peace Prize in 1989?" results in the following output:

AltaVista query:	"monetary value*" +"Nobel Peace Prize*" 1989*
Question focus:	money|numb

The L2L module converts a natural language query or question into a generic logical representation, which can be interpreted by different search engines. The conversion from language to logic takes place based on an internal query sublanguage grammar, which has been developed by CNLP. Prior to conversion, query processing such as stemming, stopword removal, and phrase and Named Entity recognition take place. We experimented with query expansion for first stage retrieval but experienced a slight drop in the results. Based on these results query expansion was left out of the TREC-9 question-answering system.

Question focus recognition aims to determine the expected answer by analyzing the question. For example, consider the question: "What is the monetary value of the Nobel Peace Prize in 1989?" The questioner is obviously looking for a monetary value and that is the focus of the question. Determining the question focus (also referred to as question type, answer type or asking point) helps to narrow the possible answers for which the system will look.

The system uses two strategies to determine the question focus: the question, and, if that strategy fails, the CNLP Named Entity hierarchy. The first strategy tries to find the focus of the question based on clues found directly in the question itself. If the beginning of a question resembles any of a set of clues it is clear what focus is intended. For example, if a question contains the words "which capital city" then the focus is "city".However, it is impossible to predict all possible questions and to have a program that deals with any question. If the system cannot assign a focus to a question using example question phrases, the system then moves to the Named Entity hierarchy clues. The system incorporates one or more clue words for each of the hierarchy classes. For example, the words *hurricane* or *storm* in a question might indicate that the questioner is looking for a weather event. The "why" focus is an exceptional case since it does not indicate a particular topic but rather a place in the sentence where an answer might be found (e.g. after the word "because"). The performance of the focus recognition capability is analyzed in section 5.3.

3.2 Document processing for first stage retrieval

We used two different retrieval approaches for first stage retrieval: Boolean and probabilistic. The entire TREC-9 question answering document collection has been indexed using AltaVista Search Engine 3.0, which is a modified version of the software that runs the search engine at http://www.altavista.com. [2] AltaVista 3.0 indexes all words and does not use stemming. The document collection consisted of 978,952 documents with the average number of words per document being less than 500. Indexing this collection took approximately 6 days using a Dual Pentium III, 550Mhz, with 512MB ram, running Windows 2000 server. AltaVista also provides the Search Developer's Kit (SDK). The SDK's Interoperability API allows programs to read data from indexes created by the search engine. A batch process takes the L2L query representations and the index directory of document collection as input. For each question, the program returns up to 32,000 documents.

For our probabilistic runs we used the SMART retrieval runs as provided by NIST. The SMART information retrieval system, originally developed by Salton, uses the vector-space model of information retrieval that represents query and documents as term vectors. [5] All vectors have *t* components where *t* is the number of unique terms (or stems) in the collection. A comparison of the Boolean and probabilistic first stage retrieval approaches can be found in section 5.1.

3.3 Paragraph finding

The system uses paragraphs rather than documents for its second stage retrieval. Based on the TREC question-answering guidelines and last year's questions, we assumed that the desired answers were going to be short and factual (less than 50 bytes long). Also, the answer context, which identifies an answer as belonging to a certain question, is usually a small part of the original document. [4] Paragraphs, which are much shorter than documents, have the added benefit of cutting down costly processing time. Paragraph detection is based on text indentations.

```
889: What is the highest mountain in the world?
Question focus: mnt (mountain)
<NC cat=numb> two|CD </NC> <CN> three-person|JJ team|NNS </CN> of|IN <NP cat=geoadj id=3>
American|NP </NP> ,|, <NP cat=geoadj id=0> Soviet|NP </NP> and|CC <CN> <NP cat=geoadj id=1>
Chinese|NP </NP> climber|NNS </CN> will|MD attempt|VB to|TO reach|VB the|DT top|NN of|IN <NC
cat=dist> 29,028-foot|JJ </NC> <NP cat=mnt id=2> Mount|NP Everest|NP </NP> ,|, the|DT world|NN
's|POS <CN> highest|JJS mountain|NN </CN> ,|, on|IN <NC cat=time> May|NP 6|CD </NC> .|.
```

Figure 3. Example of tagged paragraph (AP900429-0033) with answer "Mount Everest."

In the paragraph finding stage, we aimed to select the most relevant paragraphs from the retrieved documents from the first stage retrieval step. Paragraph selection was based on keyword occurrences in the paragraphs. The top 300 most relevant paragraphs were selected for each question. After selection, the paragraphs were part of speech tagged and categorized by <!**meta**Marker>™ using CNLP's categorization rules (see figure 3).[1] The quality of selected paragraphs and the system's categorization capabilities directly impact later processing such as co-reference resolution (currently not implemented), and answer finding.

3.4 Keyword and entity based answer finding

The keyword and entity based answer finding process took the tagged paragraphs from the paragraph finding stage and identified different paragraph windows within each paragraph. A weighting scheme was used to identify the most promising paragraph window for each paragraph. These paragraph windows were then used to find answer candidates based on the question focus. All answer candidates were weighted and the top 5 were selected.

3.4. 1. Paragraph-window identification and selection

Paragraph windows were selected by examining each occurrence of a question keyword in a paragraph. Each occurrence of a keyword in relation to the other question keywords was considered to be a paragraph window. A keyword that occurred multiple times thus resulted in multiple paragraph windows, one for each occurrence. A weight for each window was determined by the position of the keywords in the window and the distance between them. An alternative

BEST COPY AVAILABLE

weighting formula was used for single-word questions. The window with the highest score was selected to represent that paragraph. The process was repeated for all 300 paragraphs resulting in an ordered list of paragraph windows - all potentially containing the answer to the question.

3.4.2 Answer candidate identification

Answer candidates were identified in each paragraph window based on the question focus. Each paragraph window can have multiple answer candidates. If the question focus matched any of the categorized named entities, complex nominals, or numeric concepts in the window, they were considered to be answer candidates. If none of the categorized entities matched the question focus, the system translated the focus into a more general tag. For example, if the question focus called for a city and the paragraph did not have a city tag, the system then looked for a named entity in that paragraph. Naturally these matches received lower weights than entities that directly matched the question tag. If there was no question focus assigned to the question, the system reverted to an alternative strategy and picked the sentence with the largest number of question keywords and looked for named entities. In identifying the different answer candidates, the required window sizes of 50 or 250 bytes were also generated.

3.4.3 Answer-candidate scoring and answer selection

The system used a weighting scheme to assign a weight to each answer candidate. The weight was based on the keywords (presence, order, and distance), whether the answer candidate matched the question focus, and punctuation near the answer candidate. This resulted in a pool of at least 300 candidates for each query. The 5 highest scoring answer candidates were selected as the final answers for each question. The answer strings were formatted according to NIST specifications of either 50 bytes or 250 bytes depending on the run. This process was repeated for all 693 questions resulting in an answer file of 4815 (693x5) lines that were submitted to NIST.

4. Results

Our submission for the question-answering track consisted of four different runs. The SUT9bn3c runs use our L2L module (see section 3.1) with the AltaVista retrieval system for the first-stage retrieval, whereas the SUT9p2c3c runs used the SMART (provided by NIST). Each of these runs had a 50 byte as well as a 250 byte answer string submission. A system bug caused our 250 byte answers to be about 50 bytes shorter (see table 1), which caused a slight drop in results. The program only extended the number of answer bytes on the right-hand side of the answer string but failed to do so on the left-hand side.

Averages over 682 questions (strict evaluation):	SUT9 bn3c050	SUT9 p2c3c050	SUT9 bn3c250	SUT9 p2c3c250
Allowed answer length in bytes	50	50	250	250
Average response length in bytes	49.68	49.65	203.24	198.62
Mean reciprocal rank (682 questions)	0.247	0.249	0.365	0.385
Questions with no answer found	436 (63.9%)	439 (64.4%)	334 (49.0%)	319 (46.8%)
Questions above the median[1]	191 (28.0%)	190 (27.86%)	202 (29.62%)	198 (29.03%)
Questions on the median	427 (62.61%)	450 (65.98%)	351 (51.47%)	358 (55.64%)
Questions below the median	64 (9.48%)	42 (6.16%)	129 (18.91%)	99 (14.52%)

Table 1. Question answering results for all four runs.

---

[1] The median is the middle score (or the average of the two middle scores in case of an even number of scores) for each question after the answer scores for all participants have been put in rank order. 33 groups submitted a 50 byte runs, 42 groups submitted a 250 byte run.

The measure used for evaluation in the question-answering track is the mean reciprocal answer rank. For each question, a reciprocal answer rank is determined by evaluating the top five ranked answers starting with one. The reciprocal answer rank is the reciprocal of the rank of the first correct answer. If there is no correct answer among the top five, the reciprocal rank is zero. Since there are only five possible ranks, the mean reciprocal answer ranks can be 1, 0.5, 0.33, 0.25, 0.2, or 0. The mean reciprocal answer ranks for all the questions are summed together and divided by the total number of questions to get the mean reciprocal rank for each system run.

As is to be expected, the 50 byte runs have a much larger number of questions without an answer than the 250 byte runs. In all four runs, for most questions the system performance equaled the median reciprocal rank of all runs. The majority of the remaining questions were placed above the median.

Answer ranks	**SUT9 bn3c050**	**SUT9 p2c3c050**	**SUT9 bn3c250**	**SUT9 p2c3c250**
Correct answer ranked 1	126 (18.48%)	128 (18.77%)	193 (28.30%)	208 (30.50%)
Correct answer ranked 2	43 (6.30%)	42 (6.16%)	59 (8.65%)	52 (7.62%)
Correct answer ranked 3	35 (5.13%)	37 (5.43%)	45 (6.60%)	46 (6.74%)
Correct answer ranked 4	21 (3.08%)	22 (3.23%)	28 (4.11%)	31 (4.55%)
Correct answer ranked 5	21 (3.08%)	14 (2.05%)	23 (3.37%)	26 (3.81%)
No correct answer found (rank 0)	436 (63.93%)	439 (64.37%)	334 (48.97%)	319 (46.77%)
Total	682	682	682	682

Table 2. Answer rank distribution of question answering results.

The strength of our system lies in answer ranking. Consistently across all four runs, the majority of the correct answers were ranked first. Unfortunately, in all four runs we had trouble locating the answers to the questions.

5. Analysis

This section examines retrieval performance of first stage retrieval, the Language-to-Logic module, and question focus assignment as well as exact answer finding and the effect of question variants on system performance. Overall analysis based on the probabilistic 50 byte run (SUT9p2c3c050) shows that the system retrieves at least one relevant document for each of 625 questions. In the paragraph finding stage we extract paragraphs from 609 of these documents. Out of these 609 paragraphs, 578 paragraphs contain a possible correct answer. However, for only 243 questions we find that correct answer in these paragraphs. Thus, it appears that the answer scoring mechanism and entity tagging, need further refinement.

5.1 First stage retrieval

The analysis of the first stage retrieval was based on the list of relevant documents provided by NIST. We used two different first stage retrieval approaches, a Boolean approach using our L2L module with AltaVista, and a probabilistic approach using the SMART runs (see section 3.1).

Analysis shows that the retrieval performance of both systems is very similar except for the retrieved number of relevant documents, which is larger for SMART (see table 3). This difference is probably caused by a number of AltaVista query representations that had a large number of mandatory terms and failed to retrieve a single document.

Although the SMART retrieval system retrieves more relevant documents, the performance of the two first-stage retrieval models in question answering is very similar. SMART performed slightly better in the 250 byte runs (see table 1).

	Boolean	Probabilistic
Questions without any retrieved documents	3	0
Questions without any relevant retrieved documents	50	48
Questions for which relevant documents are unknown[2]	20	20
Questions with relevant retrieved documents	620	625
Total number of questions for first stage retrieval	693	693
Total number of documents retrieved	111,530	134,600
Number of known relevant documents	7,963	7,963
Total number of relevant documents retrieved	5,579	6,014
Average Precision[3]	0.2766	0.2870

Table 3. First stage retrieval performance.

5.2 Question representation

Logical question representations are one of the things created in the question processing stage (see section 3.1). The question representation analysis is based on the probabilistic 50 byte run (SUT9p2c3c050). A close examination of the question representations created by our Language-to-Logic module showed that for 539 (78.89%) questions, the representation was correct, although 64 (9.38%) representations could stand to be improved. 144 (21.11%) question representations had one or more problems. The most frequently occurring problems were: part-of-speech tagging errors; difficulties with query length (single word questions and very long questions), and; keyword selection problems (see figure 4).

Problem count	Problems with description
76	part-of-speech errors: wrong tags lead to bad phrases and non-content words being added to query
49	query length: single word queries provide little information for answer finding, long queries with many mandatory terms hinder retrieval
6	misplaced wildcards: wildcards placed on final terms of multi-word terms only, or in the wrong place of single terms creating bad stems
10	keyword selection problems: content words such as numbers erroneously filtered out

Figure 4. Question representation problems.

It is clear that the part-of speech tagger had trouble dealing with the unusual phrase structure presented by questions. Other problems, such as the single word queries, are a direct result of the phrasing of the original question. Question expansion for second-stage retrieval might be a solution for this problem. Keyword selection is an L2L problem that needs to be adjusted to keep numbers, and possibly adjectives, that specify the answer (i.e. Who was the first Russian astronaut to walk in space?).

The query representation problems were expected to have a negative impact on answer finding but further analysis showed that this was not the case (see table 4). Even with a problematic question representation, the system was still able to find answers for 77 questions while for 276 questions that did have correct query representations, no correct answers were found. This means that query representation alone only accounts for part of the error.

---

[2] Number includes the 11 questions discarded by NIST and 9 questions for which no relevance judgments were available.

[3] Average precision over all relevant documents, non-interpolated.

	Correct representation	Problematic representation
Answer correct	166 (37.56%)	77 (32.08%)
Answer incorrect	276 (62.44%)	163 (67.92%)
Total	442	240

Table 4. Question representation correctness and question answering ability.

5.3 Question focus

As described in section 3.1, we determined the focus based on the question clues or Named Entity Hierarchy clues. The question focus analysis is based on the probabilistic 50 byte run (SUT9p2c3c050). Out of 682 answerable questions, our system determined a question focus for 434 (63.64%) of the questions. Out of these 434 questions, 348 questions (80.18%) had a correct focus, and 86 questions (19.82%) had an incorrect focus. For 248 (36.36%) questions, our system could not determine a focus.

	Correct question focus	**Incorrect question focus**	**No determinable question focus**
Rank 1	97 (27.87%	5 (5.81%	26 (10.48%)
Rank 2	22 (6.32%	4 (4.65%	16 (6.45%)
Rank 3	19 (5.46%	2 (2.33%	16 (6.45%)
Rank 4	9 (2.59%	1 (1.16%	12 (4.84%)
Rank 5	7 (2.01%	1 (1.16%	6 (2.42%)
Rank 0	194 (55.75%	73 (84.88%)	172 (69.35%)
Total	348	86	248

Table 5. Answer rank distribution of question focus status.

Out of all the questions that ranked the correct answer first, 97 questions (75.78%) had a correct question focus. It appears that a correct focus aids in answer ranking. When looking at the questions with an incorrect query focus (86) we see that most of these questions (73, or 84.88%) failed to retrieve an answer at all. We can conclude that it pays to have a determinable focus as long as this focus is correct. However, finding the correct query focus is not a guarantee for finding the answer since 194 questions (55.75%) with a correct focus did not retrieve a correct answer.

A closer examination of questions with an incorrect question focus shows that 40 of these questions are erroneously assigned a "person" focus. 17 of the erroneous person focus questions are of the "who is Colin Powell" type. Unlike questions such as "who created the Muppets?" the answer to "who is <person name>" questions is not a person' s name but rather a description of that person. Additional problems with the person focus were questions looking for groups of people (i.e. cultures, sports teams) rather than individual persons, or other entities than persons (i.e. companies, cartoon characters).

5.4 Question variants

As described in section 2, NIST included 193 question variants which are re-wordings of a set of 54 questions (see figure 2). The question variants analysis is based on the probabilistic 50 byte run (SUT9p2c3c050). These question variants allowed us to study the effect of question formulation on system performance. For 25 out of the 54 question sets, the query variation caused no difference in performance. The majority of these questions did not retrieve correct answers no matter how the questions were posed to the system.

29 question sets did show differences in retrieval performance. For 12 sets, the performance differences originated entirely in additional question terms being either present or missing. For 7 sets, the differences in performance were partially due to divergence of question terms. Some question terms would guarantee a correct answer, whereas others would throw the results off. The majority of the questions are rather short, so each question term has a relatively large influence on finding the answer. The query variant results indicate that query expansion could have a large impact on system performance. Although we experimented with query expansion for first stage retrieval, we did not have enough time to explore it in the answer-finding stage.

For 14 sets, some of the differences in system performance appear to be caused by a different or missing question focus. In eight question sets, some of the differences in performance were caused by the question focus being incorrect. Additional question words mislead the system in choosing the wrong question focus. In sets where the question focus is either missing, different, or incorrect, the well-performing counterpart questions did have the correct or more exact focus, and the variant questions, without the exact clue, experienced a drop in rank or a failed attempt to find the answer. These findings indicate that having a correct question focus is of importance, which supports findings of the question focus analysis (section 5.3).

In seven question sets, some of the differences were caused by inconsistencies in the answer judgments. Certain answers would be judged to be correct for some questions, whereas for others the same answer would be judged to be incorrect.

5.5 Exact answer finding

Although plans for an "exact answer" run were abandoned by NIST, we examined the system' s exact answer-finding capabilities for the probabilistic 50 byte run (SUT9p2c3c050). The majority of the exact answers that our system produced were judged correct (197 or 81.07%), and only 46 (18.93%) of the answers were produced by the context of the answer window (see table 6). This indicates that our system had quite a high answer-finding accuracy when a correct answer was contained in the retrieved document.

Question answered at rank ...	**Number of Q. judged correct**	**Exact correct answer string found**	**Answer produced by context words in the 50-byte window**
Rank 1	128	112	16
Rank 2	42	31	11
Rank 3	37	27	10
Rank 4	22	15	7
Rank 5	14	12	2
Total	243	197 (81.07%)	46 (18.93%)

Table 6. Rank distribution of correctly answered questions and our system performance

6. Conclusions and future research

The performance of the CNLP question answering system is highly encouraging. The majority of the correct answers are ranked first and the majority of question representations and assigned question foci were accurate. The prototype system also does well at exact answer finding. However, for a large number of questions no correct answers are found. It appears that the current system does not capitalize on the large number of relevant documents found in the first retrieval stage.

Further research is needed to refine the weighting in the paragraph selection and answer finding stages, and to improve the query sublanguage grammar to increase question focus assignment robustness. In addition, a new morphological analyzer needs to be implemented and the part-of-speech tagger needs to be trained on question phrase structure, to improve question representations. A more detailed study of the categorization performance and coverage is also in order. Time also needs to be spent on researching and implementing a second approach to answer finding based on template matching.

Acknowledgements

We would like to thank Catie Christiaanse, Michelle Monsour, and Eileen Allen for creating the necessary categorization rules in a very limited time frame. We would also like to thank Hassan Bolut, and Wen-Yuan Hsiao for their engineering support. Lastly we would like to thank Lois Elmore for keeping us all in line.

References

[1] <!metaMarker>™ . http:// http://www.solutions-united.com/products_information.html

[2] AltaVista Search Engine 3.0. http://solutions.altavista.com/downloads/downloads.html

[3] Cooper, William S. (1978) The "Why Bother?" Theory of Information Usage. *Journal of Informatics*, 2, p. 2-5.

[4] Prager, John; Brown, Eric; Coden, Anni. (2000). Question-Answering by Predictive Annotation. In: *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, July 24 - 28, 2000, Athens, Greece.

[5] Salton, G. Ed. (1971) *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall, Inc. Englewood Cliffs, NJ. 556p.

[6] Voorhees, Ellen M.; Tice, Dawn M. (1999). The TREC-8 Question Answering Track Evaluation. In: E.M. Voorhees and D.K. Harman (Eds.) *The Eighth Text REtrieval Conference (TREC-8)*. 1999, November 17-19; National Institute of Standards and Technology (NIST), Gaithersburg, MD.

TNO/UT [*]at TREC-9: How different are Web documents?

[†]Wessel Kraaij, and [††]Thijs Westerveld

[†]TNO-TPD
P.O. Box 155, 2600 AD Delft
The Netherlands
`kraaij@tpd.tno.nl`

[††]University of Twente, CTIT
P.O. Box 217, 7500 AE Enschede
The Netherlands
`westerve@cs.utwente.nl`

Abstract

Although at first sight, the web track might seem a copy of the ad hoc track, we discovered that some small adjustments had to be made to our systems to run the web evaluation. As we expected, the basic language model based IR model worked effectively on this data. Blind feedback methods however, seem less effective on web data. We also experimented with rescoring the documents based on several algorithms that exploit link information. These methods yielded no positive result.

1 Introduction

The basic idea for the web track run was to modify our ad-hoc system for the main web task and perform some experiments with the link structure information. We did not know to what scale we would have to re-engineer our systems to be able to deal with 10 giga bytes of data which is about 5 times larger than the ad-hoc collection. We applied the same IR model based on an interpolated unigram language model which had proven to be succesful on several data collections and tasks: Ad hoc, CLIR, SDR and filtering. The model will be presented in Section 2.

2 Retrieval Model

All runs were carried out with an information retrieval system based on a simple unigram language model. The basic idea is that documents can be represented by simple statistical language models. Now, if a query is more probable given a language model based on document d_1, than given e.g. a language model based on document d_2, then we hypothesise that the document d_1 is more relevant to the query than document d_2. Thus the probability of generating a certain query given a document-based language model can serve as a score to rank documents with respect to relevance.

$$P(T_1, T_2, \cdots, T_n|D_k)P(D_k) = P(D_k)\prod_{i=1}^{n} \lambda P(T_i|D_k) + (1-\lambda)P(T_i) \tag{1}$$

In the above formula, $T_1, .., T_n$ represents a query, $P(D_k)$ is the a priori probability of relevance of a document. The product term consists of the probability of a term given a document $P(T_i|D_k)$ interpolated with the marginal $P(T_i)$. For the a priori $P(D_k)$ we usually take:

$$P(D_k) = \frac{\text{dlen}_k}{\sum_{j=1}^{N} \text{dlen}_j} \tag{2}$$

---

[*] The TNO and University of Twente team is a continuation of the "TwentyOne" cooperative team which participated in previous TREC evaluations.

BEST COPY AVAILABLE

This choice can be motivated by the fact that empirical studies by Singhal [4] have shown that there is usually a linear relationship between probability of relevance and document length.

The model was implemented in a vector product form supported by the TNO search engine. Our system was able to index the 10 Gigabyte dataset in roughly 20 hours on a SUN ultrasparc 300 Mhz. No re-engineering was necessary, except for the HTML entity conversion, which broke on several non-conforming documents.

3 Content only Experiments

We experimented with several variants for the estimator of the marginal $P(T_i)$ in formula (1). We compared an estimator based on the document frequency:

$$P(T_i) = \mathrm{df}_i / N \tag{3}$$

with an estimator based on the collection frequency:

$$P(T_i) = \sum_{j=1}^{N} \mathrm{tf}_{ij} / \sum_{i=1}^{V} \sum_{j=1}^{N} \mathrm{tf}_{ij} \tag{4}$$

and an estimator based on the term frequency averaged over all documents:

$$P(T_i) = \sum_{j=1}^{N} (\mathrm{tf}_{ij} / \mathrm{dlen}_j) \tag{5}$$

In these estimators, N is the number of documents and V the indexing vocabulary.

A second experiment dealt with score normalisation. Score normalisation is not necessary for the web task, but is relevant for other tasks like CLIR and topic tracking. We had found that dividing the RSV by the query length helps to normalize scores across topics. This makes sense because the RSV is composed of a sum of log terms. (cf. [3] for a description of the vector space implementation of the model, which is based on taking the log of the probability, thereby converting the product into a summation) However, when we choose a model which includes a document prior $P(D_k)$, the RSV is not a sum of query term related addends anymore, because the document prior is a constant probability, independent of the query length, which is even added when the query has zero length. We assumed that we could correct for this problem by assuming that both the document prior and the query dependent score component (the first term) are independent sources of evidence, in that case we can add a weighting component β , which controls the ratio of the prior evidence component in the final RSV.

$$RSV(Q, D_k) = 1/T_n \sum_{T_i = T_1} T_n \log(\lambda_i P(T_i|D_k) + (1 - \lambda_i) P(T_i)) + \beta \log P(D_k) \tag{6}$$

Experiments showed however, that the assumption that both sources of evidence are independent, is not true. The original model where the document prior is seen as an internal component of the model and where the sum component is not normalised separately showed the best performance. This leaves the RSV normalisation problem (which is not relevant for the web task) yet unsolved. We hypothesize that the document priors are especially helpful as an additional probabilistic knowledge source, when the system does not have a lot of information about the topic of interest (e.g. the query is short). For more informative queries, the influence of the a priori knowledge that longer documents tend to be more often significant is small, because this effect is implicitly coerced by the retrieval model. The longer the query, the lower the probability that a short document contains all query terms.

We tested several blind feedback methods on the TREC8 2 Gigabyte small web task. We did not find a consistent improvement, for title queries the performance was even hurt. We decided to refrain from feedback in the TREC9 web runs. We think the blind feedback was especially troubled by the presence of typos, which are abundant in web documents. These typos receive a high weight in most pseudo feedback strategies, because of there low document frequency. A more detailed analysis is required to study whether this is the only problem.

Table 1 gives the results of the content only runs. We have focussed on title only runs, because we feel these are most real-life and challenging.

runtag	official run	description	average precision
tnout9t2	yes	title run with 0.5 doc priors	0.1801
tnoutf1	yes	full run without doc priors	0.2178
df-estimator	no	title with doc priors	0.1871
df-estimator	no	title without doc priors	0.1465
cf-estimator	no	title with doc priors	0.1884
avtf-estimator	no	title with doc priors	0.1871
df-estimator	no	full with doc priors	0.2240

Table 1: Content-only results

The first of the official runs (tnout9t2) is a title run based on the third (average tf) estimator with a lambda value of 0.01 to enhance coordination, a prior weight β of 0.5 while dividing the first term by the query length (according to formula (6)), the second official run (tnoutgf1) is a full run based on the first estimator using the standard model of (1) without document priors. In the full run terms from the title receive a triple weight, terms from the description run receive a double weight and terms from the narrative section a single weight. This choice was motivated by some post hoc experiments on prior collections.

We have done some additional experiments. First we modified our tokenizer to allow query terms with digits to enter the fuzzy matching process. This brought a small but insignificant improvement (only one topic changed).

We also re-tested the different estimators in combination with standard document priors and different lambda values. It turned out that the choice of a lambda value of 0.80-0.90 was best for all three estimators, with very small performance differences the table shows results for lambda=0.1. The second estimator, based on the collection frequencies scored best, but practically spoken, the three estimators work about as well.

We have made some additional plots to check whether the assumption that probability of relevance is linearly correlated with document length holds for a number of collections:

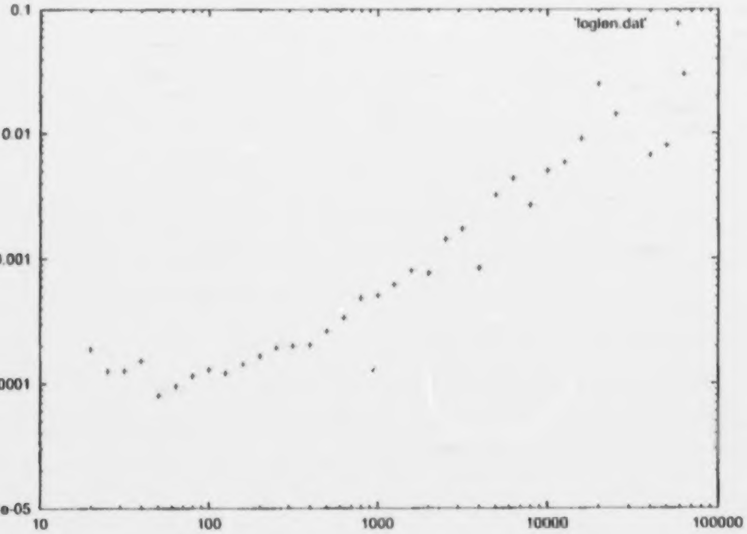


Figure 1: $P(D|l)$ for the TREC7 Ad Hoc collection

Figures 1,2, 3 and 4 show plots of $P(D_k\text{isRel}|\text{dlen}_k \in \text{bin}_k)$. Similar to Singhal, we binned the documents from

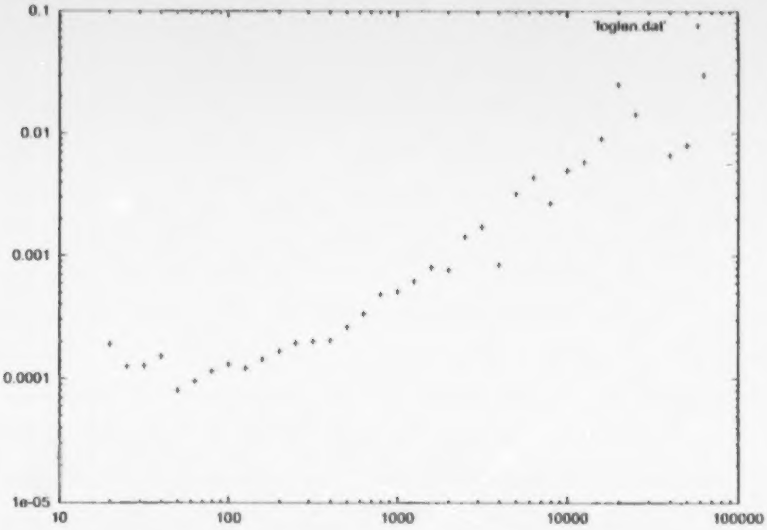


Figure 2: $P(D|l)$ for the TREC8 Ad Hoc collection

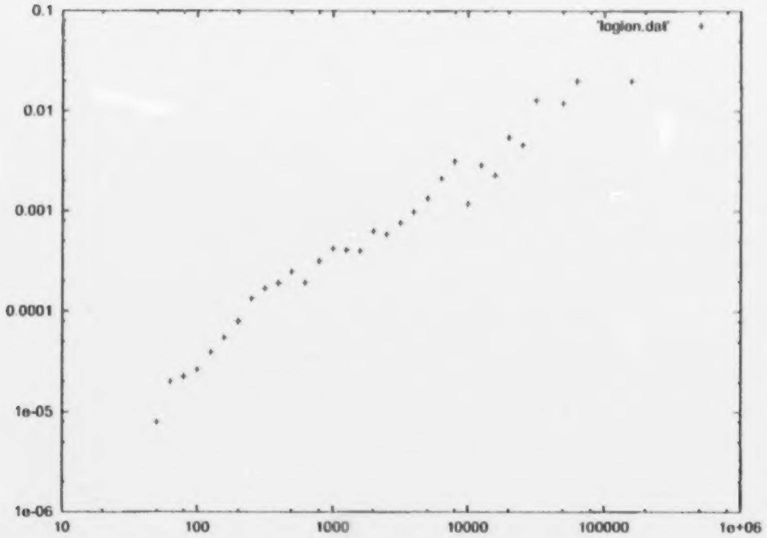


Figure 3: $P(D|l)$ for the TREC8 small web collection

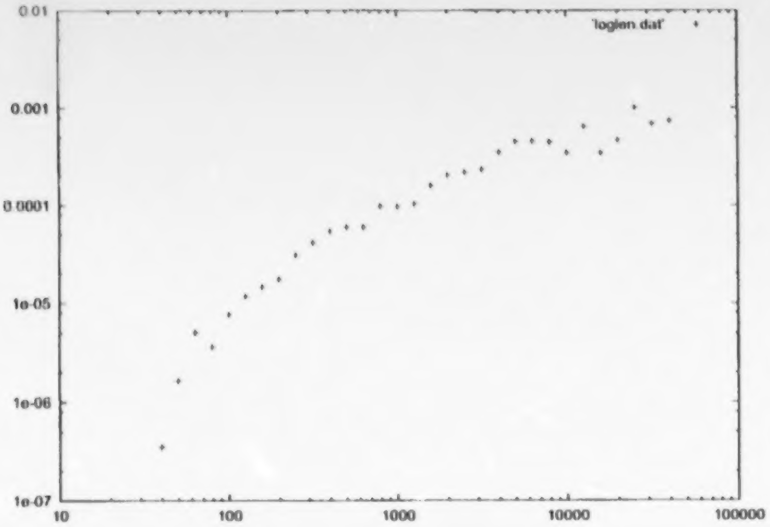


Figure 4: $P(D|l)$ for the TREC9 main web collection

the qrel file, but we took bins on a log scale. Subsequently, we computed

$$P(D_k\text{isRel}|\text{dlen}_k \in \text{bin}_k) = \frac{P(\text{dlen}_k \in \text{bin}_k|D_k\text{isRel}).P(D_k\text{isRel})}{P(\text{dlen}_k \in \text{bin}_k)} \tag{7}$$

The plots for the web collections seem quite comparable and distinct from the adhoc plots, which in turn are also quite comparable. Especially shorter Ad Hoc documents are relatively much more relevant than their web counterparts. This could be explained by the fact that shorter web documents are often just placeholders for links or pictures. We might be able to improve the performance of our runs by taking this fact into account while estimating document priors.

4 Exploitation of links

We used different link-based techniques to recalculate the scores for the top 1000 documents retrieved by a title-only, content-only run (tnout9t2). Although in last year's TREC, adding link information didn't seem to help, we hope that the higher density of links in this year's collection can improve the results. Below, we first discuss the different approaches and then compare the results to the original content-only run.

We used two different approaches. The first one is the well-known Kleinberg algorithm of hubs and authorities [2]. We took the top N documents with their in and out links and computed hubs and authoroties on that set. We then normalised the content only scores in the same way the scores are normalised in the kleinberg algorithm (equation 8). The normalised scores and kleinberg scores are then summed.

$$newscore(d) = \frac{score(d)}{\sum_{d \in docs}(score(d))^2} \tag{8}$$

The second approach is based on co-citation [5] and bibliographic coupling [1]. The assumptions behind the use of these measures to adjust the document scores are the following. If the set of documents that document A refers to is similar to the set of documents that document B refers to, then document A and B are similar. If the set of documents that refer to A is similar to the set of documents that refer to B, then A and B are similar. First we analysed last year's results. We propagated the relevance judgements along the links and computed the following scores:

$$Inlinkrel(d) = \frac{\sum_{i \in inlinks(d)} relevancy(i)}{\#inlinks(d)} \tag{9}$$

$$Outlinkrel(d) = \frac{\sum_{i \in outlinks(d)} relevancy(i)}{\#outlinks(d)} \tag{10}$$

$$Citerel(d) = \frac{\sum_{i \in inlinks(d)} Outlinkrel(i)}{\#outlinks(d)} \tag{11}$$

$$Bibcouplrel(d) = \frac{\sum_{i \in outlinks(d)} Inlinkrel(i)}{\#inlinks(d)} \tag{12}$$

In table 2 the average scores are shown for the whole collection, the assessment pool and the relevant set. Relevant documents have higher cocitation and bibliographic coupling scores than an average (judged) document. We used this information to recalculate the scores of a topic only run in the following way. We took the top N retrieved documents and propagated their scores allong their in and outlinks calculating cocitation and bibcoupling scores analogously to the cocitation and bibliographic coupling relevancies in equations 11 and 12. We used the resulting cocitation and bibliographic coupling scores to weigh the original content-only scores.

	Relevance	Inlinkrel	Outlinkrel	Citerel	Bibcouplrel
Collection (WT2g)	0.00921076	0.012829736	0.004616373	0.00728053	0.006673368
Assessment pool	0.050987634	0.010024281	0.004668967	0.010140689	0.010697012
Relevant set	1	0.064735196	0.026876365	0.126311025	0.137629731

Table 2: Average indirect relevancy

Due to some misunderstanding about the calculation of the scores, in the official content-link runs the content-only scores are reweighed by multiplying the content-only scores and the link scores (i.e. cocitation scores). However, the original content-only scores are composed of a sum of logarithms of different weights. In unofficial runs (with runtags ending in log), the link scores are properly combined with the content only scores. The results for the different runs can be seen in table 3

Adding link information decreases or at the best doesn't influence the average precision. When we take a closer look at the different link runs and compare them to the content only title run, we see that most runs hardly differ from it. The only run that differs a lot is tnout9t2lk50. In this run, the authority scores are added to the normalised

runtag	official run	description	average precision
tnout9t2	yes	title only content run	0.1801
tnout9t2lk50	yes	kleinberg on top 50	0.0488
tnout9t2lc10	yes	cocitation on top 10	0.1630
tnout9t2lc50	yes	cocitation on top 50	0.1337
tnout9t2.klein50log	no	kleinberg on top 50	0.1803
tnout9t2.coc10log	no	cocitation on top 10	0.1786
tnout9t2.coc50log	no	cocitation on top 50	0.1784
tnout9t2.bib10log	no	bibcoupling on top 10	0.1691
tnout9t2.bib50log	no	bibcoupling on top 50	0.1642

Table 3: Content-Link results

content only scores. This means that the link information is regarded equally important as the content information. In all the other runs, link information was only used to reweigh the content information. When we use kleinberg authority scores for reweighing (tnout2.klein50log), this doesn't change the results of the content run . Even though this year's collection is bigger and has more links than last year's collection, the use of link information does not seem to improve the retrieval results. One of the reasons for this might be that TREC topics are not suitable for using link information because they are too specific. This year's TREC topics on average have only 47.4 relevant documents, so the changes of being many links between them are rather small. Another reason is that there's a lot of garbage in the link information. The basic assumption behind these link based methods is that pages are topically related if they are linked to each other. Obviously, this isn't necessarily the case. Many links on the web refer to creators of the page, sponsors, friends or other pages without a topical relation to the source page of the link. Classifying links in advance into meaningful and meaningless links on the basis of for example the anchor text might help.

5 Conclusion

To our surprise, the web task turned out to be more difficult than we expected. Firstly, web documents (even though the collection has been cleaned) contain a lot of trash, often in the form of of incorrect HTML. The HTML parsing component had to be adjusted to be able to handle this kind of material. Secondly, web documents contain a lot of misspellings. These are often very rare terms. When such terms occur in a pseudo feedback document, they will have a bad influence on the pseudo feedback process, because these terms will receive a high weight. Finally, the title only queries posed a problem. Some queries contained typos, involving digits (topic 487). Our engine simply discarded those terms. The tokenizer has to be updated as well to deal with years. Four-digit years were important query concepts in quite a few queries, they were discarded by our term extraction module.

The content-only runs were finally run with some small variants of our standard IR model. Both the full and title runs perform well (33 resp 37 topics) above median, confirming the adequacy of the model. The runs which additionally analyzed link information in order to rescore the runs, were not able to improve on average precision. This confirms the general result of the TREC8 web runs. We hope to improve the link analysis in the future by looking at the anchor texts.

References

[1] M.M. Kessler. Bibliographic coupling between scientific papers. *American Documentation*, 14:10–25, 1963.

[2] J.M. Kleinberg. Authorative sources in a hyperlinked environment. In *Proceedings of 9th ACM-SIAM Symposium on Discrete Algorithms*, pages 668–377, 1998.

[3] W. Kraaij, R. Pohlmann, and D. Hiemstra. Twenty-one at TREC-8: using language technology for information retrieval. In *The Eighth Text Retrieval Conference (TREC-8)*. National Institute for Standards and Technology, 2000.

[4] Amit Singhal, Chris Buckley, and Mandar Mitra. Pivoted document length normalization. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29, 1996.

[5] H. Small. Co-citation in the scientific literature: A new measure of the relationship between documents. *J. American Soc. Info. Sci.*, 24:265–269, 1973.

A semantic approach to Question Answering systems

Vicedo, Jose Luis & Ferrández, Antonio.
{vicedo,antonio}@dlsi.ua.es

Dpto. Lenguajes y Sistemas Informáticos. Universidad de Alicante
Campus de San Vicente del Raspeig
Apartado 99. 03080 Alicante, Spain

Abstract

This paper describes the architecture, operation and results obtained with the Question Answering prototype developed in the Department of Language Processing and Information Systems at the University of Alicante. Our approach accomplishes question representation by combining keywords with a semantic representation of expected answer characteristics. Answer string ranking is performed by computing similarity between this representation and document sentences.

1 Introduction

The prototype presented in this paper tries to face up question answering task from a new point of view. Question analysis obtains a mixed representation of queries based on keywords and a semantic representation of main information characteristics required by the question. Sentence ranking algorithm combines both representations to rank and select the best five answers. In the following section, our system is described. Afterwards, we analyse results obtained in TREC-9 Question Answering task. Initial conclusions are extracted and finally, directions for future work are discussed.

2 System Overview

Our system is structured into two main modules: *Question analysis module* and *Answer selection module*. First module processes questions expressed in open-domain natural language in order to obtain a representation of the information requested. This analysis is accomplished by obtaining *question type* and classifying terms into *keywords* and *definition terms*. Keywords help the system to locate sentences where answers can probably be found. A term in a query is considered a definition term if it defines characteristics of the expected answer. Question type and definition terms define the main information required by each question. A WordNet-based tool process questions type and definition terms in order to obtain a semantic representation of expected answer characteristics. This representation defines what we call *semantic context* of the target answer. The answer selection module uses keywords and semantic context to locate the sentence containing the answer and extract the part of the sentence that contains it. Figure 1 shows system architecture.

BEST COPY AVAILABLE

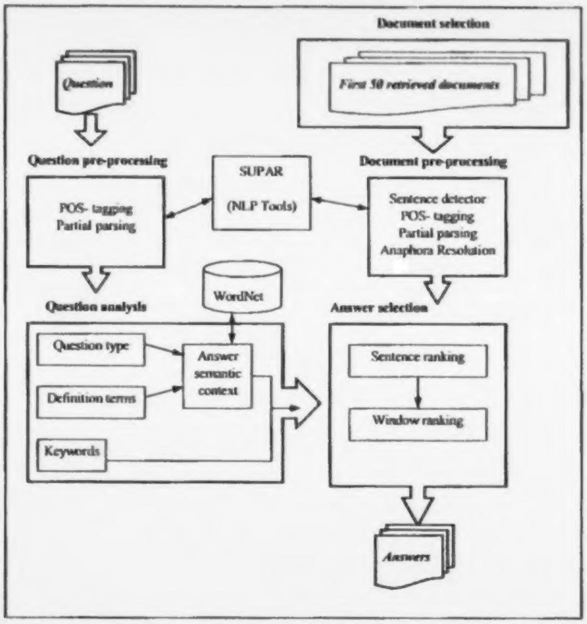

Document selection
Question
First 50 retrieved documents
Question pre-processing
SUPAR
(NLP Tools)
Document pre-processing
POS- tagging
Partial parsing
Sentence detector
POS- tagging
Partial parsing
Anaphora Resolution
WordNet
Question analysis
Answer selection
Question type
Answer
semantic
context
Definition terms
Keywords
Sentence ranking
Window ranking
Answers

Figure 1. System architecture

2.1 Document Selection

We only processed the first fifty ranked documents supplied by TREC Organisation. Nevertheless, all QA track collection was analysed to obtain term *idf* weights (Salton, 1989). Term normalisation was performed using a version of Porter's stemmer.

2.2 Question and Document pre-processing

Several Natural Language Processing techniques have been applied to both questions and documents. These tools compose the Slot Unification Parser for Anaphora Resolution (SUPAR) described in Ferrández (1999, 1998). SUPAR's architecture consists of three independent modules that interact with one other. These modules are lexical analysis, syntactic analysis, and a resolution module for NLP problems such as anaphora resolution. Queries and documents are pre-processed before entering question analysis and answer selection modules. Queries pre-processing consists on part-of-speech-tagging terms and parsing. Documents are pre-processed by detecting sentence boundaries, part-of-speech-tagging terms, parsing and solving pronominal anaphora. Managing with pronominal anaphora consists on substituting non-pleonastic third-person pronouns for their antecedents.

BEST COPY AVAILABLE

2.3 Question Analysis

Question processing module accomplishes different tasks. This module extracts main keywords, expands keyword terms, determines question type and builds the semantic context representation of the expected answer.

Question type is detected by analysing Wh-terms. This process maps Wh-terms into one or several of the following categories:

PERSON GROUP LOCATION TIME
QUANTITY REASON MANNER NONE

Each of these categories is related to WordNet top concepts (Miller, 1995). When no category can be detected by Wh-term analysis, NONE is used (e.g. "What" questions). This analysis gives the system three kinds of information: (1) lexical restrictions that expected answer should validate, (2) how to detect *definition terms* (if they exist), and (3) top WordNet concepts relevant to the expected answer.

Definition terms do not help the system to locate the correct answer but instead, they usually describe the kind of information requested. Depending on question type, different approaches are used to detect definition terms. For "What", "Which", "How", and similar questions these terms are detected by selecting noun phrases appearing next to the Wh-term. When questions such as "Find the number of..." or "Name a flying ..." are analysed, noun phrases following the verb are considered definition terms.

Once question type and definition terms are analysed, the system generates the *semantic context of the expected answer*. A WordNet-based tool processes each definition term in order to build its semantic context representation. This context is represented as a weighted term vector that is computed as follows: for each definition, synonyms, one-level search hyponyms and all hyperonyms (until a top concept is achieved) are obtained. The weight assigned to these new terms is the *idf* of the analysed definition term in the collection divided by the distance in the WordNet hierarchy from this term to each new obtained one. When question type has been successfully mapped to a top concept, only terms related to this concept will be added to the term context representation. This way we obtain the terms that made up the context of a unique definition term. The semantic context representation of the answer (the joined representation of all definition term contexts) is computed by adding the context vector of each definition term in the question.

The semantic context of the answer helps the system in different ways: First, it approximates the type of the expected answer when the Wh-term analysis has been unable to obtain it. Second, as top concepts are too broad, it allows sub-classifying them for each particular question. And third, it helps the system to decide between different possible answers by comparing expected answer and probable answer semantic contexts.

To finish with question analysis, remaining question terms are considered keywords. When there are no remaining terms left (e.g. for the question "Name a flying mammal"), definition terms are used as keywords too. Non proper noun keywords are expanded using WordNet by adding to the question, keyword synonyms, one-level search hyponyms and one-level search hyperonyms.

2.4 Answer Selection

The input to this module is a small number of pre-processed candidate documents and the results of question analysis module. As first step, sentences are ranked accordingly to the following score:

$$Sentence\text{-}score = Keyword\_idf\_sum + (0.65 * Expanded\_keyword\_idf\_sum)$$

where :

Keyword_idf_sum: is the sum of the *idf* weights for query keywords that appear in sentences.
Expanded_keyword_idf_sum: is the sum of the *idf* weights for terms obtained when expanding query keywords that also appear in sentences.

In both cases, the *idf* of a term that occur twice or more times in a sentence is added only once.

When this initial sentence ranking has finished, the first 100 ranked sentences that include probable answers are selected as the best candidates to contain the correct one. A term is considered a probable answer if it verifies lexical restrictions obtained by Wh-term analysis.

The final step is to analyse sentences to extract and rank the windows of the desired length that probably contain the correct answer. The system selects a window for each probable answer by taking as centre the term considered a probable answer. Each window is assigned a *window-score* that is computed as follows:

$$Window\text{-}score = Sentence\text{-}score * (1+\cos(Question\_SC, Window\_SC))$$

where :

cos: Cosine
Question_SC: vector representing the semantic context of the expected answer.
Window_SC: vector representing the semantic context of terms contained into the selected window (excluding keywords and expanded keyword terms).

Finally, windows are ranked on window-score and the system returns the first five ranked windows as final result.

3 Results

TREC-9 Question Answering Track allowed five answers for each question. Besides, depending on answer-string length, two different run types were defined: up to 50 or 250 bytes long. We participated with two runs for each different answer length. Figure 2 shows results obtained. ALI9C runs have been produced applying the whole system described above. ALIC9A files contain results obtained applying the same strategy but without solving pronominal anaphora in relevant documents. These results were computed after getting rid of eleven questions whose answer did not appear in the document collection. Therefore, only 682 questions were evaluated.

Run	Answer lenght	Mean reciprocal rank		% Answers found	
		strict	lenient	strict	lenient
ALI9C250	250	35,6%	37,1%	52,9%	55,3%
ALI9A250	250	34,9%	36,3%	51,6%	53,8%
ALI9C50	50	23,0%	24,5%	33,9%	36,1%
ALI9A50	50	22,7%	24,0%	33,9%	35,8%

Figure 2. QA Track results

Although a detailed results analysis is a very complex task, several conclusions can be extracted.

Retrieving relevant documents.
Correct answer was not included into the top fifty ranked documents supplied by TREC for 95 questions. As this fact relies on document retrieval strategy, we can not measure how our approach managed with these queries. Figure 3 analyses the percentage of questions that could be correctly answered depending on the number of top documents selected for searching the answer. Even if first 1000 documents were analysed, it would have been impossible to obtain the correct answer for 25 questions. It seems that document retrieval techniques do not fit QA retrieval needs. In fact, systems applying paragraph-indexing techniques (Harabagiu, 2000) (Clarke 2000) have obtained a better performance.

Top Docs Selected	10	25	50	100	250	500	750	1.000
Answer included	499	547	587	607	629	641	650	657
Answer Not included	183	135	95	75	53	41	32	25
% Answer Included	73,2%	80,2%	86,1%	89,0%	92,2%	94,0%	95,3%	96,3%

Figure 3. Document retrieval analysis

Context based answer detection.
Our main objective was to inspect how Wh and definition terms could be used to build a useful semantic representation of expected answer and if this representation could improve correct answer detection. Results analysis shows several circumstances. This approach increases system performance by comparing expected answer with probable answer contexts. Very good results are obtained when possible answers context gives some indication about the nature of these answers. In this case context analysis allows the system to find the correct answer, even to successfully decide between similar but different possible answers. However, when possible answer context does not include characteristics that define the possible answer, the system does not take profit of expected answer context definition. It seems clear that semantic context representation can not substitute the use of a Name-Entity tagger (not applied in our prototype). We think that combining both tools will contribute to improve system performance in two important aspects: (1) increasing the amount and quality of the information obtained from the question and (2) improving possible answers detection.

Another circumstance to take into account is the way of selecting terms that define the context of the possible answers. Nowadays, the system builds the semantic context of a possible answer from all terms included into the window (250 or 50 bytes) surrounding each probable answer. As results show, results have become poorer as answer length decreased. This fact relies on the number and type of terms selected for building possible answer semantic context.

Pronominal anaphora resolution

Application of pronominal anaphora resolution has produced only a small benefit (around a 1%). Analysing this fact is very difficult but it relies on two main reasons. First, we have noticed that the number of relevant sentences involving pronouns is very low. And second, there are a lot of documents related to the same information, and sentences in a document that contain the right answer referenced by a pronoun, can also appear in another document without pronominal anaphora. Anyway, although the benefit is low, it can be considered a blind evaluation of how automatic pronominal anaphora resolution always helps QA systems performance.

4 Future Work

Several areas of future work have appeared while analysing results. First, IR system used for retrieving relevant documents has to be adapted for QA tasks. The IR used by TREC Organisation retrieved the document containing the correct answer into the first fifty relevant documents only for a 86% of the evaluated questions. Second, question analysis has to be improved by increasing the number of question types analysed (i.e. definition or list questions). Third, unless context based answer detection has revealed to help system performance it needs a finer tuning on defining the number and type of terms used for semantic context building and exploring the possibilities of a Name-Entity tagger. This strategy needs to be investigated and tested.

5 References

Clarke C., Cormack, G., Kisman D. and Lynam T. (2000) *Question Answering by passage selection*. In Proceedings of the Nineth Text Retrieval Conference. Washington (USA).

Ferrández A., Palomar M. and Moreno L. (1998) *Anaphora resolution in unrestricted texts with partial parsing*. In Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, COLING – ACL. Montreal (Canada).

Ferrández A., Palomar M. and Moreno L. (1999) *An empirical approach to Spanish anaphora resolution*. To appear in Machine Translation.

Harabagiu S., Moldovan, D., Pasça M., Mihalcea R., Surdeanu M., Bunescu R., Gîrju R., Rus V., and Morărescu P. (2000) *FALCON: Boosting Knowledge for Answer Engines*. In Proceedings of the Nineth Text Retrieval Conference. Washington (USA).

Miller G.(1995), *"Wordnet: A Lexical Database for English"*, Communications of the ACM 38(11) pp 39-41.

Salton G.(1989), Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer, Addison Wesley Publishing, New York.

TREC-9, 2000. Call for participation Text Retrieval Conference 2000 (TREC-9). http://trec.nist.gov /cfp.html.

The PISAB Question Answering System

Giuseppe Attardi and Cristian Burrini
Dipartimento di Informatica
Università di Pisa - Italy
{attardi,burrini}@di.unipi.it

Abstract

The PISAB Question Answering system is based on a combination of Information Extraction and Information Retrieval techniques. Knowledge extracted from documents is modeled as a set of entities extracted from text and by relations between them.

During the learning phase we index documents using the entities they contain. In the answering phase we exploit the index previously built in order to focus the search for the answer to just the most relevant documents. As answers to a question we select from these documents the paragraphs containing entities most similar to those in the question.

PISAB has been submitted to the TREC-9 Conference, achieving encouraging results despite it current prototypical development stage.

Introduction

The problem of finding answers to questions on a large document collection, could in principle be solved by creating a knowledge base with the information extracted from documents and then querying such knowledge base. Unfortunately this approach is not yet feasible, since it requires advanced techniques of natural language processing, knowledge extraction, knowledge representation and reasoning, which are beyond the current state of the art.

On the other hand, Information Retrieval techniques are quite effective in retrieving documents relevant to a certain subject, so in particular those which might contain the answer to a question. Information Extraction techniques help identifying certain kinds of information, but their capabilities are quite domain dependent and limited to entities with predefined patterns. Neither of these techniques is sufficient to address the Question Answering problem, but we have explored a way of combining them to build a complete question answering system.

The approach

The meaning of a document might be expressed in terms of entities and relations between them. Entities are the semantic equivalent of nouns present in the document, while relations correspond to verbs. For example, the information contained in the following phrase:

"John reads a book"

can be represented by means of the entities "John" and "book", and by the relation "reads" that links subject and object. Relations need not be binary: prepositional phrases and various kinds of syntactic adjuncts allow expressing *n*-ary relations.

Intuitively an entity refers to an object in the real world or to a concept, or in general to an element of the semantic domain of the document contents. In this semantic domain entities have attributes and are related to other entities [Attardi 86]. Within documents only syntactic representations of the entities are present, expressed in one of several forms:

- Proper nouns: **Microsoft**;
- Pronouns reference: Microsoft … **It**;
- Descriptions in terms of attributes values: **the world largest software company**;
- Descriptions in terms of relations among entities: **the Redmond company**.

Suitable natural language processing techniques are available for extracting entities expressed in the first two ways. The case of entities referenced by proper nouns is referred in the literature as *Named Entity extraction*. The case of pronouns is called *coreference* and *anaphora resolution*. Techniques for dealing with objects expressed by means of attributes or relations descriptions are less common.

Our system tries to locate the syntactic boundary of each kind of entity expression and, in the first two cases, it attempts to solve the references. The architecture of our system consists of an *Entity Tagger*, which performs a superset of the functions of a Named Entity tagger, and a *coreference module* (of still limited capabilities in our prototype).

Entity Tagger

The Entity Tagger works mainly at the syntactic level, exploiting features provided by lexical analyzers, like part of speech tags, cases, gazetteers and contexts rules.

For example, given the phrase "The TREC-9 Conference was held in Maryland in November 2000" the Entity Tagger produces:

< TREC-9 Conference> was held in <Maryland> in <November 2000>.

The Entity Tagger is capable of dealing with **structured entities**. Numeric dates, Web addresses, numbers, monetary expressions are examples of structured entities, which are expressed according to specific syntactic rules. Suitable Information Extraction modules are invoked in a pre-processing stage of the Entity Tagger in order to recognize structured entities.

Semantic Tagger

Having determined the syntactic boundaries of an entity, the system tries to classify it according to a predefined semantic ontology, i.e. it tries to assign to each expression the proper semantic category of the referred object.

For example, given the phrases:

<Washington> is in <North America>.

<George Washington> didn't like <apples>.

<Washington> threatens <Iraq> to start <the war>.

the semantic tagger is capable of disambiguating the three senses of the term "Washington" within each phrase, classifies each occurrence accordingly, and produces:

[Washington/LOCATION] is in [North America/LOCATION].

[George Washington/PERSON] didn't like [apples/FOOD].

[Washington/ORGANIZATION] threaten [Iraq/ORGANIZATION] to start [the war/ACT].

To perform classification, the semantic tagger exploits a thesaurus and a set of *context rules*, which allow inferring the semantic category of any entity (not just of a Named Entity). The semantic ontology used for classification is a two level tree extracted from the *is-a* relation of the WordNet [WordNet] thesaurus.

$Chunk_1{:}T_1,\ Chunk_2{:}T_2,\ \ldots\ Chunk_n{:}T_n$
AND
$Attribute\text{-}Conditions(Chunk_1,\ldots,Chunk_n)$
$\rightarrow add\text{-}Score(Chunk_k, sem\text{-}Vector)$

Figure 1 - Structure of a Context Rule

$C_1{:}NamedEntity,\ C_2{:}Verb,\ C_3{:}Entity$
AND
$(C_2.HeadWord.rootForm=\text{"be"}\ AND$
$C_3.semantic\_approximation=semx)$
$\rightarrow add\text{-}Score(Chunk_1, semx)$

Figure 2- The is-a context rules

In its general form, a context rules states that if the context surrounding an entity matches a certain syntactic pattern and certain conditions on semantic attributes of the entities involved are met, then the context provides evidence that the entity belongs to a certain semantic category, expressed as a numeric weight. For example, the rule in Figure 2, applied to fragment "John is the chief director of ...", states that the entities <John> and <the chief director> most likely have similar meanings. Entity classification is probabilistic based: the semantic tagger computes the likelihood that an entity belongs to each semantic category, and selects the category with the highest likelihood.

The pair of an entity instance and its semantic category makes what we call a *concept*, for instance ⟨George Washington, PERSON⟩.

Concept Indexing and retrieval

Our QA system is based on a concept-oriented indexing and search engine which stores the concepts extracted from documents using IE techniques. The underlying intuition is that both the documents relevant to a question and the question itself must be about the same semantic objects. Therefore documents are indexed according to the semantic entities extracted from them, and they are searched then through to the semantic entities extracted from queries. We expect the search will match only a small set of documents, strictly related to the question, within which to focus further search for the answer, since it is most likely that they contain such answer.

This document-filtering step has also efficiency benefits, since it reduces the number of documents that must be considered in subsequent steps.

Architecture

The overall architecture of PISAB Question Answering System is illustrated in Figure 3:

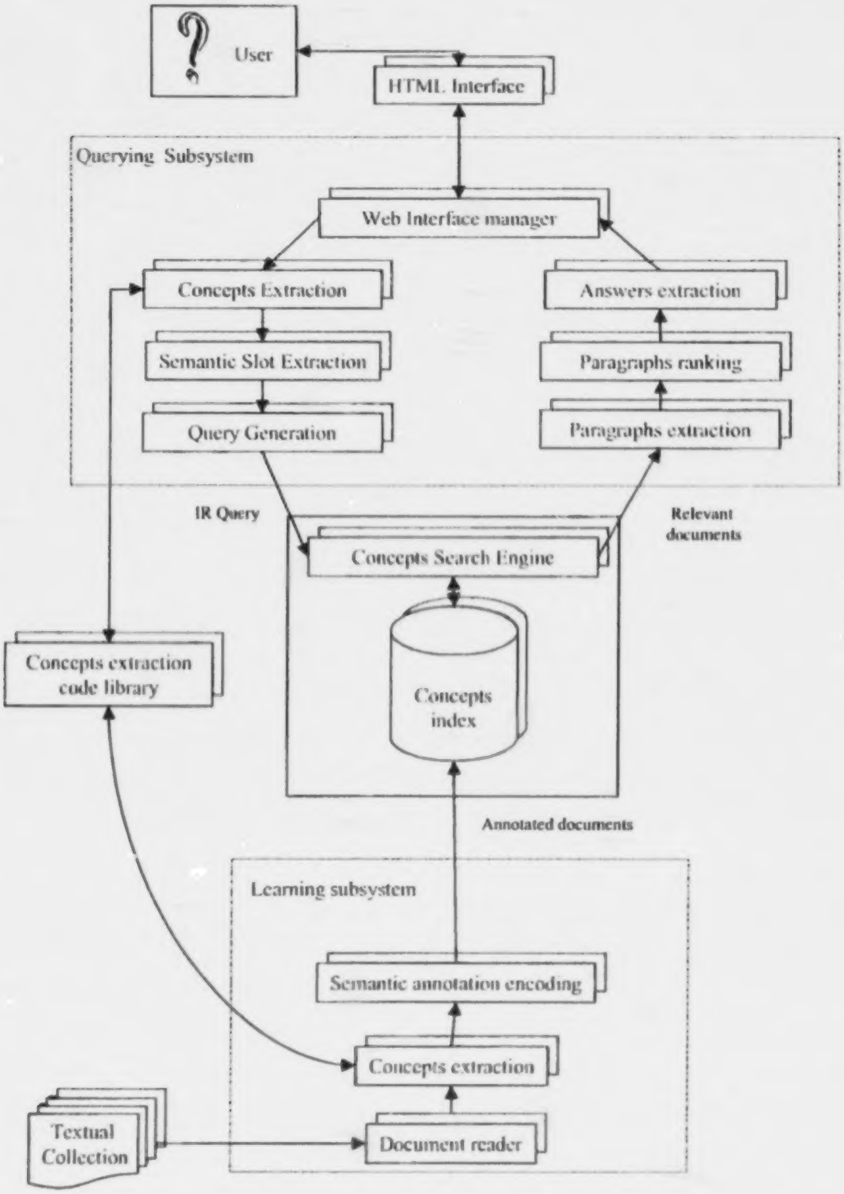

User
HTML Interface
Querying Subsystem
Web Interface manager
Concepts Extraction
Semantic Slot Extraction
Query Generation
Answers extraction
Paragraphs ranking
Paragraphs extraction
IR Query
Relevant documents
Concepts Search Engine
Concepts index
Concepts extraction code library
Annotated documents
Learning subsystem
Semantic annotation encoding
Concepts extraction
Textual Collection
Document reader

Figure 3 – Architecture of PISAB Question Answering System.

The learning subsystem extracts concepts from documents (i.e. semantic classified entities), by means of the semantic tagger. Concepts are used as indexes for the analyzed documents in a document retrieval system.

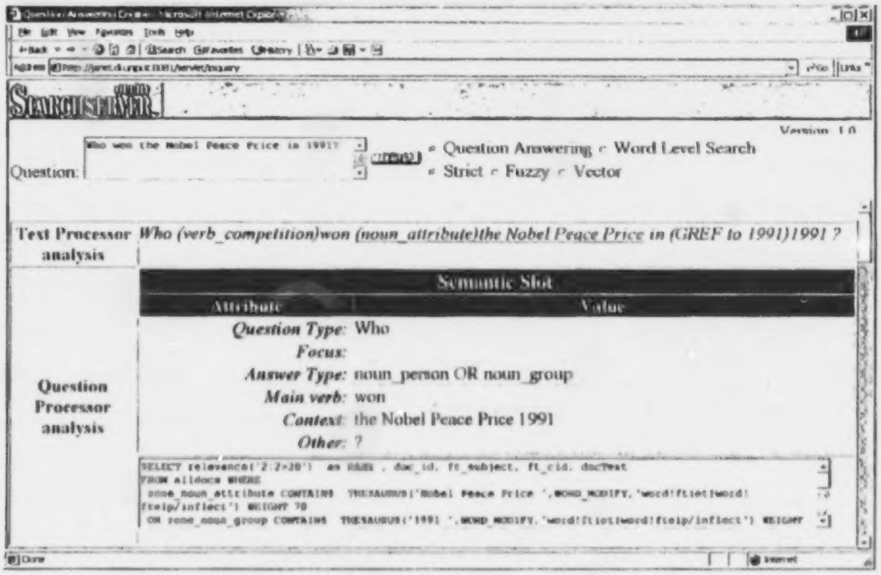
The first step in the answering phase is to extract concepts from the question. Then a query for the document retrieval system is constructed, made up from these concepts, but with different weights assigned to each concept according to its role in the question. For example, a concept that seems to be the focus of the question is weighted more than one that is not. (The focus is a concept that describes

BEST COPY AVAILABLE

semantically the answer entities. For example: in "What U.S. President did ..." the *Question Focus* is "U.S. President" and the *AnswerType* is PERSON).

This is done by analyzing the question and building an abstract representation of it consisting of several *semantic slots*, including:

- Question type: who/where/when/which/how, ...
- Main verb
- Description of the concept to find (*Focus*)
- Semantic category of the candidate answer (*Answer type*)
- Other contextual concepts (*Context*)



Question: Who won the Nobel Peace Price in 1991?

● Question Answering ○ Word Level Search
● Strict ○ Fuzzy ○ Vector

Text Processor analysis *Who (verb_competition)won (noun_attribute)the Nobel Peace Price in (GREF to 1991)1991 ?*

Question Processor analysis

Semantic Slot	
Attribute	Value
Question Type:	Who
Focus:	
Answer Type:	noun_person OR noun_group
Main verb:	won
Context:	the Nobel Peace Price 1991
Other:	?

```
SELECT relevance('2:2=20')  as RANK , doc_id, ft_subject, ft_cid, docText
FROM alldocs WHERE
 zone_noun_attribute CONTAINS  THESAURUS('Nobel Peace Price ',WORD_MODIFY,'word!ftiet|word!
fteip/inflect') WEIGHT 70
 OR zone_noun_group CONTAINS  THESAURUS('1991 ',WORD_MODIFY,'word!ftiet|word!fteip/inflect') WEIGHT
```

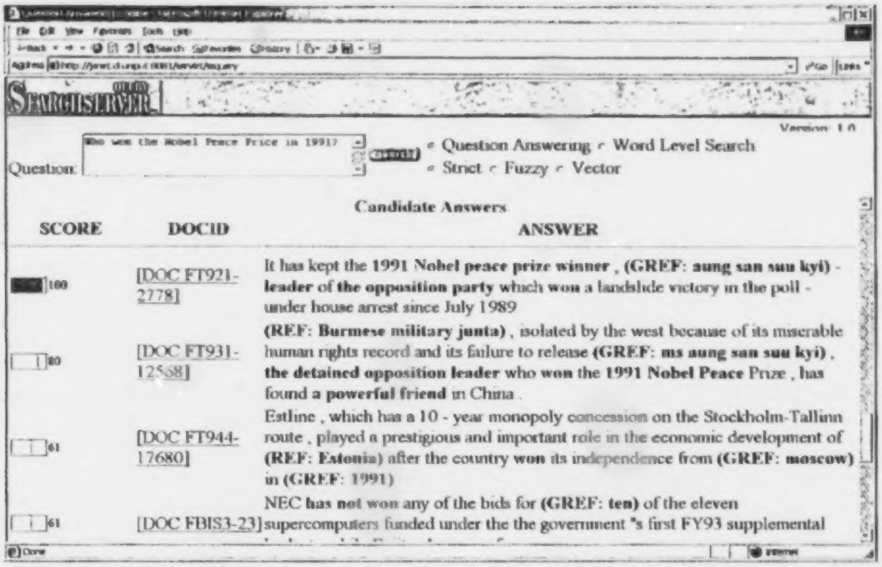
Figure 4 PISAB prototype Web Interface.
Shown are: question, choice of IR search method, results of NLP analysis of question, filled semantic slots and sketch of generated query.

To increase the robustness of document relevance ranking we combine concept weighting with a traditional IR scoring function: the *cosine distance* between the text and the query. Adding such term-related score mitigates the effects of errors in entity extraction and classification. The query is finally expanded by means of a thesaurus, including variants of concepts and terms present in the question, for improving matching likelihood.

The expanded query is used to retrieve documents relevant for the question. The most relevant ones, according to the search engine rank, are further analyzed in order to extract the candidate answers. We split the document into paragraphs (or sentences) and rank them according to an estimate of relevance to the query. We have a concept occurrence, also called a "semantic hit", when a paragraph contains an inflection of a question's concept or an entity with the same semantic category of the Question Focus Each occurrences is weighted according to the associated semantic slot and all weights are added up to obtain a score for each paragraph.

Finally, from each top ranked paragraph we extract the best scoring text window: this will be one of the candidate answer to be displayed to the user, as shown in Figure 5.

BEST COPY AVAILABLE



Question: Who won the Nobel Peace Price in 1991?

● Question Answering ○ Word Level Search
● Strict ○ Fuzzy ○ Vector

Candidate Answers

SCORE	DOCID	ANSWER
100	[DOC FT921-2778]	It has kept the **1991 Nobel peace prize winner , (GREF: aung san suu kyi) - leader of the opposition party** which **won** a landslide victory in the poll - under house arrest since July 1989
80	[DOC FT931-12558]	**(REF: Burmese military junta)** , isolated by the west because of its miserable human rights record and its failure to release **(GREF: ms aung san suu kyi)** , **the detained opposition leader** who **won** the **1991 Nobel Peace** Prize , has found **a powerful friend** in China .
61	[DOC FT944-17680]	Estline , which has a 10 - year monopoly concession on the Stockholm-Tallinn route , played a prestigious and important role in the economic development of **(REF: Estonia)** after the country **won** its independence from **(GREF: moscow)** in **(GREF: 1991)**
61	[DOC FBIS3-23]	NEC **has not won** any of the bids for **(GREF: ten)** of the eleven supercomputers funded under the the government "s first FY93 supplemental

Figure 5 – Candidate Answers.

References

[Attardi 86]	G. Attardi and M. Simi, A Description Oriented Logic for Building Knowledge Bases, *Proc. of the IEEE*, Vol. 74, N. 10, 1986, 1335-1344.
[Lasso]	D. Moldovan, S. Harabagiu, M. Pasca, R.Mihalcea, R. Goodrum, R. Girju, V. Rus. Lasso: A Tool for Surfing the Answer Net. Department of Computer Science and Engineering Southern Methodist University. Dallas, 1999.
[LTG]	A. Mikheev, C. Grover, M. Moens. Description of the LTG system used for MUC-7. HCRC Language Technology Group, University of Edinburgh, 1998.
[PACLING]	S. Baluja, V.O. Mittal, R.Sukthankar. Applying machine learning for high performance Named-Entity extraction. PACLING'99, Waterloo, Canada, 1999.
[Textract]	Rohini Srihari, Wei Li. Question Answering Supported by Information Extraction. Cymfony Inc, Williamsville CA, 1999.
[TREC8]	Ellen Voorhess, Dawn M. Tice. The TREC-8 Question Answering Track Evaluation. National Institute of Standards and Technology, Gaithersbugh, 1999.
[VIE]	K. Humphreys, R.Gaizauskas, H. Cunningham, S. Azzam. VIE Techincal Specification. Department of Computer Science and Institute for Language, Speech and Hearing (ILASH) University of Sheffield, 1998.
[WordNet]	G.A Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11), 39–41, 1995.

BEST COPY AVAILABLE

Goal-Driven Answer Extraction

Michael Laszlo, Leila Kosseim, and Guy Lapalme
RALI, DIRO, Université de Montréal
CP 6128, Succ. Centre Ville, Montréal (Québec) Canada, H3C 3J7
`{laszlom, kosseim, lapalme}@iro.umontreal.ca`

Abstract

We describe the structure and functioning of an answer-extraction system built from the ground up, in only three man-months, using shallow text-processing techniques. Underlying these techniques is the attribution to each question of a *goal type* serving to characterize the outward form of candidate answers. The goal type is used as a filter during long-answer extraction, essentially a small-scale IR process which returns 250-byte windows rather than whole documents. To obtain short answers, strings matching the goal type are extracted from these windows and ranked by heuristics. TREC-9 performance figures show that our system has difficulty dealing with brief, definition-based questions of the kind most likely to be posed by users. We propose that specialized QA strategies be developed to handle such cases.

1 System Overview

In the following we shall refer to our system by the working title XR^3, which stands for *eXtraction de Réponses Rapide et Robuste*, or "fast and robust answer extraction". This choice of name reflects the idea that in QA, answers are not known beforehand, but are sought in raw text with the aid of hints supplied by the user. That, at any rate, was the assumption which arose from our acquaintance with the TREC-8 test set; more recent experience with the TREC-9 questions suggests that answer generation from a knowledge base will indeed be an indispensable component of future QA systems.

XR^3 is composed of two distinct halves: the first returns long answers consisting of exactly 250 characters, while the second extracts short answers consisting of no more than 50 characters. The unified view in Figure 1 shows that the second round of processing recycles output from the first, and that they share certain intermediate results.

Input to XR^3 consists of a question and a small collection of source documents. Regardless of whether these documents are paired *a priori* with the question or identified on the fly, they are expected to contain a valid answer with high likelihood. For the purpose at hand, we used Amit Singhal's lists of relevant documents (as determined by an IR process which used the question as a query). We deem these to be of sufficient quality in light of our finding that 97% of the top-200 documents per TREC-8 question contain at least one valid answer. Time constraints prevented us from implementing our own IR process, but for the sake of broad coverage we used all 1000 documents per question issued for TREC-9.

A pattern-matching analysis of the question identifies a set of *keyterms* (keywords and keyphrases), a goal, and possibly a date, all of which are used in the extraction and scoring of 250-character windows. The best five of these may be directly submitted in the long-answer category, and that is exactly what we did for TREC-9. Internally, we refer to this as the *IR run*, since our scoring heuristic employs as a primary criterion the occurrence of keyterms. The best 100 of these windows are funneled to the second portion of XR^3, although not all of them will be used. Depending on the value of a system parameter, 10 to 20 are skimmed off the

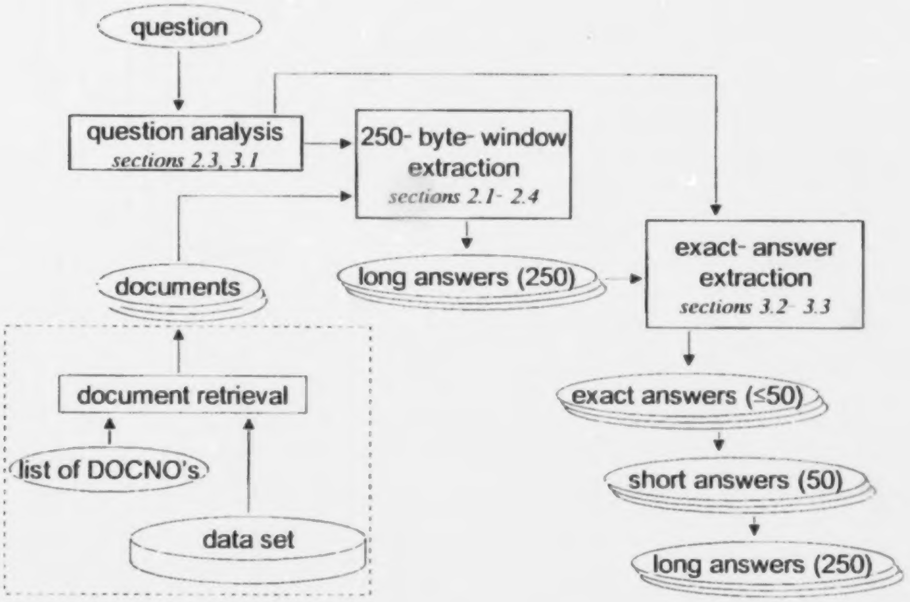

question
question analysis
sections 2.3, 3.1
250- byte- window
extraction
sections 2.1- 2.4
exact- answer
extraction
sections 3.2- 3.3
documents
long answers (250)
document retrieval
exact answers (≤50)
list of DOCNO's
short answers (50)
data set
long answers (250)

Figure 1: Process flow in XR^3: short answers are extracted from long answers; some results of question analysis are shared.

top (we settled on 10 prior to the TREC-9 runs) and, in a crucial application of the question goal which was identified earlier, exact answers are extracted. These strings are guaranteed to contain no more than 50 characters, but are typically a fifth that long, being precise expressions which name "Mount Everest" and "four miles" and other things considered to be potential answers. In any event, they are ripe for submission in the short-answer category. Expanding the string to a full 50 characters, and then to 250 characters, yields a second run in each category. In sum, then, we submitted four runs to TREC-9: two in the long-answer category, and two in the short-answer category (although one of these consists of our exact answers).

2 Long Answers

Let us note that the needs of QA are nearly satisfied by information retrieval systems, but for two shortcomings. The sheer size of a document returned by IR means that it is unlikely to fall within reasonable estimates of what constitutes an answer. And then there is no guarantee that this document will be relevant to some question on the user's mind, nor does the IR task require that a document be of any value to the user.

The quickest way around the first of these obstacles is to abandon orthodox conceptions of the document as an intact, coherent narrative. If we redefine the "document" as a string found in some portion of the corpus and not exceeding, say, 250 characters in length, then an IR system would, formally, also be a QA system. The second difficulty is the thornier one, but it is safe to say that if we were in possession of a function which maps questions to useful queries — if, that is, we had some idea of what words and phrases tend to occur in the vicinity of the answer to a given question — then IR alone would be adequate to the QA task.

These are the very principles by which our system extracts long answers. The first portion of XR^3 may be described as a lightweight IR engine adapted to the requirements of QA, having been furnished with a front-end utility which translates questions into queries and with shallow heuristics at the back end for scoring its results.

2.1 Windowing

We henceforth consider the data set not primarily as a collection of documents, but as a series of text *windows*, which are defined in XR^3 as follows. Given a list of keyterms, a window is any sequence of 250 characters found within a single document and containing a keyterm as near its center as possible. Of course, the focal keyterm should be precisely centered in all but the extremal windows, comprising the first and last 250 characters of a document, where it may be situated to the left or right of center, respectively. Apart from these two special cases, which may take hits from several keyterms, no window should be extracted more than once under our regime. In a more pragmatic sense, however, we are bound to end up with a great deal of redundancy. A non-extremal cluster of keyterm occurrences in the source text will result in the extraction of several windows that differ from one another by only a few words: these windows are semantically more or less equivalent, and would tend to receive identical scores.

The prospect of having our final list of top 50 or top 5 windows cluttered by superfluous windows is a disconcerting one, so we impose a limit on the amount of overlap permitted between any two windows in the form of a system parameter, called the maximum windowing margin, which is set at the beginning of a run and stays constant thereafter. With a margin of 0.1, for instance, we would take each cluster of windows whose offsets (distances from the beginning of the document) vary by fewer than $0.1 \times 250 = 25$ characters, arbitrarily choose one, and discard the rest.

We have been unable to settle on an optimal value for this margin, nor is it clear that there is one at all. It stands to reason that a very restrictive margin value such as 0.1 leads to the loss of viable windows, while high values such as 0.9 are insufficiently discriminating. But for the broad middle range between 0.2 and 0.8, the give-and-take of these two opposing trends, along with the perturbations resulting from the variation of other system parameters, make it impossible to arrive at an informed decision. Seeing this, we resolved to hold the margin constant at the unexceptionable value of 0.5, without pausing to wonder how many valid answers were falling through the cracks.

2.2 Keyterm Extraction

The process by which XR^3 assembles a query for its IR phase stands on the premise that words appearing in a question should, in the source text, be found in proximity to a valid answer. Our approach may be unique in that selected question tokens are used verbatim in the search query, without taking into account synonymy or even morphology. Our aversion to the use of synonyms issues at least in part from the report of the University of Ottawa's team at TREC-8, which notes that the results of their own brute-force QA system suffered from synonym sensitivity [ML99]. Furthermore, synonym occurrence is not necessarily valuable in early-phase extraction. The intuitive argument in favor of synonymy is that verbs and adverbs used in relation to an answer string are far more susceptible to variation than are the nouns, and certainly the proper nouns, found in a question. But expanding the IR query with sense-related words is likely to result in a large number of extraneous windows: if a question asks for the identity of someone who "won" the Nobel Prize, then the presence within a window of such words as "acquire", "gain", and "accomplish" may be misleading rather than useful.

Precisely because verbs and adverbs found in the question are subject to such a wide variety of rephrasings, we ignore them altogether, presuming that it is sufficient to collect all windows which make reference to, say, the Nobel Prize. In general, we regard the presence of some noun in the question as a qualifying condition for the presence of an answer. With that, we lose much of the need for stemming or morphological variation. Once those tokens tagged as determinants and prepositions and so forth have been discarded, only nouns and proper nouns, these latter having been identified on the basis of capitalization, are admitted into the search

goal type	occurrences
PNOUN	124
TIMEPOINT	23
CARDINAL	22
UNKNOWN	13
LINEAR	7
MONEY	6
PERCENTAGE	2
TIMESPAN	1
MEANS	1
REASON	1
AREA	0
VOLUME	0
MASS	0

Table 1: TREC-8 goal frequencies — number of times each type of goal was identified by the final version of XR^3 among questions 1-200 of the TREC-8 test set.

query. A second pass over the question adds groupings of adjectives, participles, and nouns, in order to capture such potentially useful noun phrases as "world energy output" and "managing director".

2.3 Goal Identification and Extraction

A further condition windows should meet is that they each contain at least one expression corresponding to the *goal* of the question, which we understand as a loose characterization of the answer. For example, if it is clear from the form of a question that one is being asked to name a river — perhaps because it reads "Name a river which ..." — then we accept no window unless it contains something that could conceivably be the name of a river, such as a proper noun. Thirteen goal types are currently distinguished by XR^3, as listed in Table 1. A goal should have the dual virtues of being easily identified in the question and easy to extract from source text. Identification is accomplished by a series of regular expressions which rely on the presence of trigger words ("Who ..." → `PNOUN`) and patterns ("How much ... earn/spend/cost/..." → `MONEY`). If a question runs the gauntlet without drawing fire, the goal type is assumed by default to be `PNOUN`, which we take to be a fairly safe bet for questions of the type seen in TREC-8.

Extraction from the source text of strings matching the goal is again performed by regular expressions. Hand-crafted patterns and trigger words, such as lists of unit names associated with `MONEY` or with `TIMESPAN`, are applied throughout, except for the extraction of proper nouns. Here we apply a novel method which begins by extracting all capitalized words and groups thereof appearing beyond the first word of a sentence. The less trivial cases, where we must consider the very first word of a sentence, are resolved by comparing that word to known prepositions, or, in the extremity, by comparing it to proper nouns that have already been identified in the middle of a sentence.

As a final criterion, windows must match the date (if any) specified by the question. This highly specialized tactic was developed in response to those 11% of questions from the TREC-8 test set which named a specific year, as in *Which team won the Super Bowl in 1968?*(16).[1] In order for a window to be eligible under these circumstances, it must either contain the specified date (permitting some variation, such as '68 for 1968) or it must be drawn from a document published on that date.

---

[1] After each question we parenthetically indicate its number, from 1 to 893, in the conventional TREC order.

score	recall	group bonus	type factor	date factor
.527	91%	0	1	1
.529	91%	0	1	10
.557	93%	0	10	1
.562	90%	1	1	1
.562	91%	1	1	10
.563	93%	0	10	10
.583	93%	1	10	1
.592	93%	1	10	10

Table 2: Parameter optimization — automatic evaluation on runs obtained by varying values of three key scoring parameters; performed only on questions 1-100 of the TREC-8 test set.

2.4 Scoring Long Answers

The answer-bearing potential of a window is measured first of all by the number of keyterms it contains — or rather by the variety of keyterms, since we do not take into account multiple occurrences of the same term. Weight values may range from 0 (so that the term is effectively ignored) or 1 (no greater weight than ordinary terms) to arbitrarily high values such as 20 (which would mean that such terms are worth as much as 20 ordinary ones). The sum obtained from the linear combination of keyterm hits (0 or 1) with their respective weights is then multiplied by some constant if the window was determined to suit the question goal, and by a final constant awarded for date correspondence. To be quite precise, the score is calculated as follows. Given keyterms $K_1, \ldots, K_n$, we know of two corresponding sequences: $c_1, \ldots, c_n$ where c_i is `capBonus` if K_i is capitalized, 1 otherwise; and $g_1, \ldots, g_n$ where g_i is `groupBonus` if K_i is a phrase, 1 otherwise. The function $f(K_i)$ returns 1 if K_i is present in the window under consideration, 0 otherwise. The partial score is given by

$$\sum_{i=1}^{n} f(K_i) c_i g_i$$

This score is then multiplied by a goal factor and a date factor as necessary. Experiments show that it is worthwhile to set these factors to a value high enough so that windows not meeting the goal and date criteria effectively drop out of sight in the final ranking. Increasing both factors from 1 to 10 improves the TREC score by a margin of 0.03. This gain, if not spectacular, is significant and strikingly consistent regardless of fluctuation in system parameters.

2.5 Results of Long-Answer Extraction

In making the quantitative observations above, we rely on TREC scores determined by an automatic evaluation which applies the extended set of Perl patterns derived from last year's judgments, producing an inflated estimate of the true score, *i.e.*, that which would be awarded by a human jury. Once system parameters have been fully tuned, XR^3 achieves a score of 0.511 on questions 1–200, which compares favorably to the results posted by last year's participants. We do, of course, possess the considerable advantage of having seen the questions beforehand. However, let it be noted that all pattern-matching rules were written for the first 100 questions, and all system parameters were optimized for the same set. Evaluation on the remaining questions gave us no reason to make changes of any significance to the IR portion of XR^3. Furthermore, automatic evaluation consistently yields TREC scores over 0.5 on any subset of questions save for the last 25, where performance suffers badly, dropping well below 0.4. We attribute this hiccup to the far greater brevity of the last 25 questions, which contain only 3.2 keyterms on average, compared to 5.7 for the set as a whole.

link type	example
`existence`	*Who is the Queen of Holland?*(136)
`attribute`	*How tall is the Matterhorn?*(161)
`time`	*When was Yemen reunified?*(130)
`location`	*Where is Inoco based?*(20)
`reason`	*Why are electric cars less efficient in the north-east than in California?*(159)
`means`	*How did Socrates die?*(198)

Table 3: Link types — the six types of question link identified by XR^3, with a characteristic example of each; the link is understood to be a relationship between the named focus and an unnamed answer.

3 Exact Answers

We go to the trouble of extracting self-contained answers from a shortlist of 250-character windows not only for the sake of submitting them in a new category, but with the hope of improving performance in the old one. It turns out that the prospects for finding exact answers are fairly bright, considering the density with which valid answers occur near the top of our long-answer list. The first five windows contain at least one valid answer about two-thirds of the time according to our automatic evaluation on the TREC-8 test set, while the number-one window alone is hit with 40% probability.

At the heart of our exact-answer extraction process is the assignment to each question of a goal, which previously played so marginal a role in window filtering. Now, however, for a given set of windows we know of a set of substrings which are of the form required by the question under consideration. The goal identification and extraction processes were designed very conservatively, emphasizing recall rather than precision; thus, we can state with reasonable confidence that if any one of the windows contains a valid answer, then one of our strings will constitute a valid answer.

3.1 Question Analysis

We have forsaken the computationally expensive and mostly impossible task of formal parsing in favor of something more modest, something that captures less information but stands a better chance of being fully realized. Our pet theory this year was that all questions explicitly describe a relationship between something unknown — to wit, the answer — and something quite concrete. The "something unknown" is already described by our goal, so it remains to identify the *link* and the *focus*. To us, the focus of a question is that portion of it which must absolutely be found within a window, whether verbatim or in some variant form, if we are to believe that this window can contain a valid answer. It is not clear whether every question contains such an expression, but we will proceed on the assumption that it is there. Given the question *What was the monetary value of the Nobel Peace Prize in 1989?*(2), for instance, the focus should be identified as "Nobel Peace Prize"; in the source text, we would then look for strings bearing some lexical resemblance to it.

A question's link is not so much identified as interpreted from its syntactical structure. Of the six possible link types (see Table 3), the one most relevant to the above question is presumably `existence` — that, at any rate, is the truth according to XR^3. It might be argued that `attribute` would be a more intuitively correct interpretation, but such quibbles are beside the point. The utility of a question analysis is sharply limited by the accuracy of the source-text analysis against which we intend to compare it, and as things stand, this latter is so much more difficult a task that even our very rough characterization of the question will suffice.

The link and focus are both determined through pattern matching. We view the question as a sequence of phrases hinging on stopwords, which are here defined as prepositions, determinants,

question	components	content of window
Which team won the Super Bowl in 1968?(116)	*pre*	terback, Namath, one of the first since Babe Ruth to make a fortune playing a game. With Namath as their leader, the AFL's 1968
	answer	New York Jets
	post	went into
	focus	Super Bowl
	right	III as an 18-point underdog and won, 16-7, against the NFL champion Baltimore Colts, wh
How many people live in the Falklands?(101)	*left*	everything. If they bought someone out, where would they go? Mr Di Tella denied that payments would be made to encourage people to leave the
	focus	Falkland Islands
	pre	and settle elsewhere. He said, 'We want to be very respectful of these
	answer	2,000
	post	people. They have liv

Table 4: Schematization of candidate windows — the text of each 250-character window is broken up into five named fields, here separated by carriage returns; note that in the first case, the answer occurs earlier than the focus, but the contrary is true in the second case.

pronouns, and common verbs such as "is" and "does". These stopwords are perceived as falling naturally into certain patterns which betray the semantic relationships between the adjoined phrases.

3.2 Window Schematization

Once XR^3 has decomposed a question with the aid of stopwords and so inferred its goal, link, and focus, it attempts to unify this description of the question with syntactical structures found in the source text. We assume that all viable windows will contain something resembling the question focus (which we shall simply call a "focus" in the interest of brevity). Furthermore, the window must contain at least one candidate answer (hereafter, an "answer") — an expression matching the question's goal. We then take the string lying between the focus and the answer, and try to ascertain whether it is related to the question link. That is not to say that the remaining portions of the window are of no interest; nonetheless, we restrict our attention to this particular string for the sake of simplicity and computational tractability.

In order to carry out such an assessment, XR^3 considers every possible focus-answer pairing found in a given window, and for each of these the window is subdivided into five components in one of the two ways illustrated in Table 4. If the answer is to the left of the focus, the appropriate scheme is [*pre, answer, post (inter), focus, right*], whereas the converse calls for [*left, answer, pre (inter), focus, post*], where *pre* and *post* are understood in relation to the answer.

3.3 Scoring Exact Answers

The overall score awarded to an answer and its attendant apparatus is a linear combination of three partial scores. The first consists of the IR score previously determined for the window from which this answer was extracted. The second component, our ***link score***, is calculated by one of six groups of regular expressions according to the value of the question link. These are triggered by patterns which have proven, in our experience, to be strongly associated with the link. In the case of existential links, one of the more prominent trends involves constructions of

run	score
long1	.511
exact	.331
short	.386
long2	.524

Table 5: Best results obtained by automatic evaluation on TREC-8 questions 1-200.

the form ***answer, a/an/the*** ... *focus*, as in "Colin Powell, an American general" or "Everest, the world's highest peak". Thus, we reward those answer schemes where the connecting string begins with a comma and a determinant.

The third component is something called the *extra score*, which is determined by ancillary considerations. Foremost among these is a special treatment for those questions whose goal has been determined to be some number of specific units, such as `CARD(calories)`. Also taken into consideration are the presence of words found in the question but not in the focus (as a precaution against focusing error) and additional hits scored by a simple stemming mechanism based on regular expressions.

3.4 Answer Expansion

Once all the goal-type expressions extracted for a given question have been scored and ranked we can offer the top five as exact answers, even though there is no such category in the TREC-9 evaluation. These strings tend to be rather shorter than the 50 characters allowed in the short-answer category — about 10 characters long — so we may easily cast our net a little wider by adding characters to the left and right until the limit is reached. The strings which result from such an expansion tend to suffer from the same redundancy problem experienced in the first phase of long-answer extraction: very often, several 50-character windows will end up covering the same cluster of exact answers. Again, XR^3 makes use of a maximum windowing margin to discard superfluous answers. This is done without disturbing the order of the remaining answers, so that they remain ranked in a useful way, namely, that determined by the exact-answer score. The top five may then be offered as a second submission in the short-answer category.

The expansion to long answers is accomplished with even greater ease, since we have retained the 250-character windows from which the exact answers were extracted. Each of these is now awarded the score of the corresponding exact answer, the entire collection is sorted, redundancies are cast out, and the result is an ostensibly improved list of long answers.

3.5 Results of Short-Answer Extraction

The limited selection of scoring criteria described above was distilled from a far broader field. Most speculative heuristics fell by the wayside as they failed to boost scores in the context of XR^3's performance on the TREC-8 test set. Among these were functions which measured resemblance between the designated focus and the candidate focus; calculated the average distance of all keyterms from the candidate answer; and employed a more comprehensive stemming algorithm than the one currently in use. In the end, we arrive at the scores displayed in Table 5. Recall that the first run was produced by means of our 250-character windowing mechanism, while the remaining three all derive from exact-answer extraction.

Our score of 0.331 for exact answers seems to be a decent performance: it would, in theory, have ranked fourth out of the 18 short-answer runs at TREC-8. Note that these exact answers were extracted from only the 10 best long answers. Any more than that and scores suffer, presumably because the scoring algorithm does fairly well at reshuffling high-quality windows but is too naive to recognize low-ranking junk for what it is. Using only the IR score to rank

run	TREC-9 assessment	+/− from mean	% from mean	mean over all runs
long1	.352	+.002	+.01%	.350
exact	.149	−.069	−32%	.218
short	.179	−.039	−18%	.218
long2	.366	+.016	+4.6%	.350

Table 6: Official results achieved on questions 201-893 of the TREC-9 set, with comparison to arithmetic mean of all runs submitted in the respective category (50 or 250 characters). Recall that our exact answers are 10 characters long on average, and that the first run is our IR run.

exact answers, rather than the entire tripartite formula, results in a not much worse TREC score of 0.237, demonstrating the pervasive influence of brute-force IR.

Sheer chance also has a role to play, given that the expansion to 50 characters results in a markedly improved TREC score of .386. Final expansion to 250-character windows leads to an expected — but unexpectedly small — improvement over the first run, from .511 to .524. Surprisingly, these TREC scores remain fairly even across the entire range of goal types, whereas one might have expected the frequent and coarse `PNOUN` extraction to fare poorly against the more specialized types, such as `MONEY` and `SPECTIME`.

4 Evaluation

4.1 TREC-9 Performance

The first column of figures in Table 6 gives the "strict" TREC-9 assessment of each of our runs, while the last column indicates the mean of the TREC scores assessed over all runs in each category. Clearly, XR^3 has suffered a general and sizeable decrease in performance — this year's long answers result in a lower score than last year's short answers. We attribute this in part to the many features of this year's test set which were not present at TREC-8.

Questions 201 through 893 tend be much shorter than the first 200, offering 3.8 keyterms on average as opposed to 5.7 with the earlier set. In addition, a notable number are of the form *What is X?*, whereas these were entirely absent from the TREC-8 set, with the exception of *What is Head Start?*(115). It is not clear that such requests for information as *Who is Anubis?*(222) and *Where is the Danube?*(226) fit the description of "short-answer, fact-based questions," as the QA Track Guidelines would have it. What is clear, however, is that these are precisely the sorts of questions that users want to ask and do end up asking today, even in queries submitted to commercial search engines.

We noted an unusual number of misspellings among the questions: *Superbowl*, *Nicholas Cage*, *applicances*, and *Pittsburg*, for instance, are all erroneous, and a system which made no attempt at error correction, such as ours, had to suffer the consequences. There was also a high incidence of reverse question construction, such as *Colin Powell is most famous for what?*(835), as well as a heavy reliance on synonym-based back-formation, especially among the last 193 questions. We were, of course, given ample warning that these features would be present, and made some effort to construct regular expressions for the decomposition of reverse-syntax questions.

4.2 Discussion

Our results demonstrate the viability of goal-augmented IR as a preliminary phase in QA, with several important qualifications. First, given that the brute-force IR portion relies on collocation alone, this approach will provide good answers only for questions which provide good clues. Second, the goal identification and extraction mechanism is much better at excluding worthless windows than at seeking out promising ones. In the most pragmatic terms, this means that we can expect fairly high recall (80 to 90% for the top 100 long answers) but fairly low precision (30 to 40% at the first rank). Further and deeper NLP seems to be in order.

It appears that the engineering problems of QA, concerning such things as corpus management, text cleaning, and retrieval of fixed-size passages on the basis of keywords and keyphrases, are largely resolved in our implementation, and that what remains is the fundamental question of how an automatic process can be made to recognize the correspondence between a question (***Who was President Cleveland's wife?***(11)) and its answer ("... Grover Cleveland married Frances Folsom..."). Questions, due to their limited syntactical complexity, are relatively easy to deal with. The most difficult problem in QA remains the analysis of source text, and a good way to handle it seems to be the goal-driven approach, which isolated small, answer-rich snippets of text rather than attacking the whole.

4.3 Future Work

We suggest that a high-performance QA system can be built from a more fine-grained scheme of goal identification and extraction than our current, rather rudimentary one which has achieved a degree of success commensurate with its small scope. It would be best to begin with a subcategorization of proper nouns into such sharply delimited concepts as tall buildings, popular brand names, and famous composers. This would, in effect, be a taxonomy of named entities conceived for the purpose of QA. The extraction of such highly-specialized named entities requires that we have on hand a collection of exhaustive word lists. We propose to compile a wholly new collection by an automatic process performed on a large conventional dictionary.

In addition, we are excited by the prospect of broadening the reach of QA — and perhaps increasing its precision — by opening the floodgates to the Internet. A simple HTTP interface to an online search engine would permit the same sort of information-retrieval processes which our system currently applies on the TREC corpus to be used on documents drawn from the world over. It remains to be seen whether the Web yields useful text passages, but it would be interesting to experiment with the use of various sources, including the "hardwired" corpus, various online works of reference, and the Internet at large, in order to see whether answer confidence may be increased through the correlation of multiple occurrences.

5 Conclusion

Our experiments show that a textual pattern-matching mechanism, properly engineered, can produce satisfactory results on verbose questions of the kind found in the TREC-8 test set. The appeal of such an approach is further amplified when we consider its ease of implementation and modest computational demands. The reason for its success is transparent: in order to elicit a very specific answer, users are often compelled to provide a context-rich question. We have demonstrated the viability of collocation in a full-fledged QA system when it is bolstered by the specialized processes of goal identification and goal extraction. These processes, furthermore, are prerequisite to the extraction of exact-answer strings. It is equally clear that pattern matching is less effective in dealing with terse questions and with those which ask for a definition. In our view, the increasing heterogeneity of questions will have to be met by increasingly specialized QA methods. In particular, we conjecture that definition-based questions are best answered from works of reference, and that one of the vital ingredients in general-purpose QA is an array of tightly constrained goal categories.

References

[ML99] Joel Martin and Chris Lankester. Ask Me Tomorrow: The University of Ottawa Question Answering System. In *Proceedings of TREC-8*, pages 575–583, Gaithersburg, Maryland, 1999.

The system RELIEFS : a new approach for information filtering

Christophe Brouard and Jian-Yun Nie

Laboratoire RALI
Département d'Informatique et de Recherche Opérationnelle
Université de Montréal
CP. 6128 succ. Centre Ville
Montréal QC H3C 3J7, Canada

{brouardc,nie}@iro.umontreal.ca

Abstract

In this year's filtering track, we implemented a system called RELIEFS that tries to learn about the prediction capability of words or conjunctions of words for the relevance of documents. The novelty of the system resides in two main points. First, the features used in the prediction involve both : the implication D->Q (from document to query), and the reverse implication Q->D. This is different from usual approaches where only the first of the implication is used. Therefore, the relevance estimation of a document combines the probability that a document containing a term is relevant, and the reverse probability - the probability that a term appears in relevant documents. The second novelty is that, in addition to the use of words as prediction elements, we also consider word combinations (conjunctions). However, not all combinations are significant. Therefore, an incremental algorithm is developped to select only the meaningful conjunctions. To limit the number of conjunctions, we do not use a cut on conjunction length. Rather, we eliminate the conjunctions A&B that bear the same information as A or as B. Our first results prove the feasibility of the approach. Other experiments are ongoing in order to fully evaluate this approach.

1. Introduction

The goal of our participation in TREC9 is to experiment the following two ideas for information filtering :

The first idea is about the use of the two implications D->Q (from document to query) and Q->D. Usually, in Information Retrieval, relevance evaluation is based on the evaluation of D->Q (van Rijsbergen, 1986). If one considers a document as a set of terms, and a query as a specification of what we are looking for, the implication D->Q may be decomposed to the judgment of "if the term is present then the document is relevant" for each term of the document.

Even if some authors signal the importance of the reverse implication (Q->D) (Nie, 1988), the relation has not been integrated in relevance evaluation. This relation has been taken into account in probabilistic models as a way to calculate D->Q. In our approach we will consider both implications simultaneously. the consideration of the reverse implication Q->D means in practice that we have to consider the relation "if the document is relevant then the term is present in the document". From a pragmatic point of view, the use of Q->D may be justified by the fact that it allows us to favour terms which have been met many times in relevant documents, comparing to rare terms for which the presence in relevant document is a coincidence. The two implications may be illustrated as two relationships between terms and relevance as in Fig 1. The two relationships are different in nature and both are important for judging the relevance of a document. Therefore, we will integrate both of them in our approach.

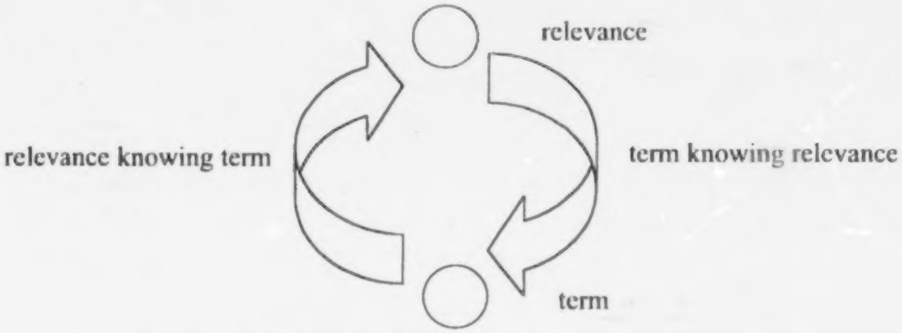


Figure 1 - Relationships between terms and relevance

The second important aspect in our approach is the use of term combinations. Usually, the learning for adaptive filtering system consists of updating the weights of terms and not term conjunctions. This is because the assumption that terms are independent. It is also due to the fact that considering term combinations would lead to a combinatory explosion. Some methods tried to consider term combinations, but usually limited themselves to a certain length. This solution is not totally satisfactory since the length constraint can not be completely justified. Morever, the number of combinations is still very high. We propose here to update all the implications whithout loosing any information. The economy principle we propose is based on the observation that if two terms t_1 and t_2 are always present simultaneously (in the same documents), it is useless to create the information $t_1\&t_2$ since this information is the same as t_1 (or t_2). In this way, many combinations can be eliminated. We use an incremental algorithm (Brouard, 2000) to determine whether a combination should be added and its weight be updated.

2. System description

The goal of RELIEFS system is to find words or conjunctions of words that are good predictors of document relevance. The RELIEFS processing can be decomposed in three steps: 1. Selection of N document words from the document, 2. Estimation of the document's relevance, 3. Revision of word's predictability.

2.1 Step 1 : Selection of N document words

All the document words are compared with the words which have been extracted from the query, the document examples given for learning and the documents which have been previously selected. The considered words or word conjunctions are elements of prediction p_i. They are sorted by the value of their predictability of relevance. This predictability is estimated as the product between the relative frequency of relevance knowing p_i and the reverse frequency, i.e. $F(R/w_i).F(w_i/R)$. If less than N words (in our experiments we choose N=20) can be selected in this way, this selection is completed first by the document words which are related to the query words and finally by the document words in their lecture order. The relatedness between words is estimated using both implications on the training set (Ohsumed 87). In our solution of additional words, those that are related to sereval query words are given priority.

2.2 Step 2 : Evaluation of the relevance

Considering the elements of prediction which appear in the document, the score of the document is computed as follows :

$$\frac{\sum_{i^*=1}^{k} F(R/p_{i\bullet}).F(p_{i\bullet}/R)}{\sum_{i=1}^{n} F(R/p_i).F(p_i/R)}$$

where $F(R/p_i)$ is the relative frequency of relevance given the presence of the element of prediction p_i in this document, $F(R/p_i)$ is the reverse relative frequency and i* are the indices of the elements of prediction which are present in the document. In RELIEFS, the relevance of a document is estimated as the sum of the implication products for all the elements of prediction present in the document divided by the sum of the implication products for all the elements of prediction. In the example of Fig 2, word5 and word8 are elements of prediction and appear in the document, the implication products of these elements are taken into account and increase the score of the document.

2.3 Step 3 : Updating relative frequencies

If the evaluation of step2 is larger than a defined threshold, then the N words selected in the first step are submitted to an updating process on their relative frequencies, and new conjunctions are also built. The building condition of a conjunction A&B is that F(A/B) and

F(B/A) are different from 1. This condition allows us to avoid building useless conjunctions (i.e A&B is equivalent to A or/and to B).

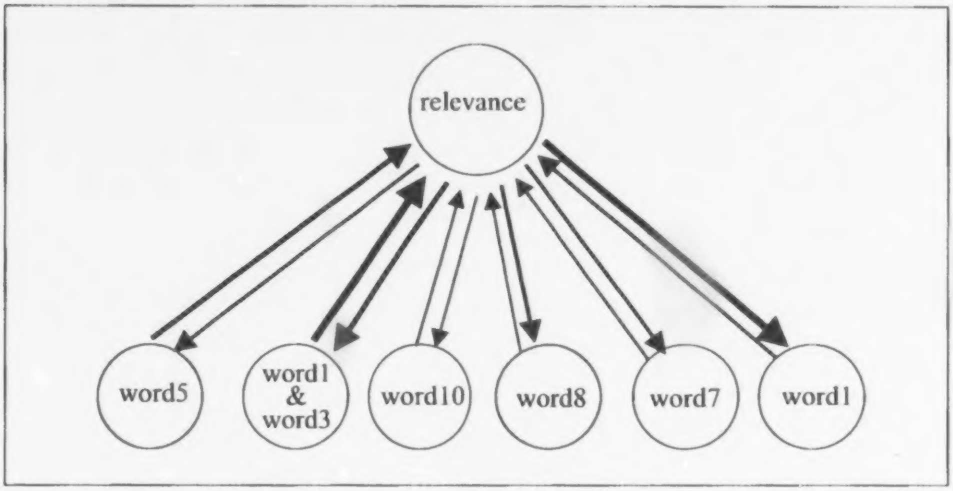


Figure 2 - Relevance evaluation in RELIEFS

2.4 Threshold adaption

We tried to adjust the thresholds with a very simple mechanism. When a selected document is irrelevant, the threshold is increased with a small value. Each time a document is not selected, the threshold is decreased. The initial threshold is computed on the basis of the score of the two first relevant documents and the amplitude of the threshold modification is based on the difference between the average score of the two last relevant documents and the two last irrelevant documents. Initially, we considered an average of 0 for irrelevant documents. So the change tends to be larger at the begining than at the end. Morever, the product of the change scale by a constant allows us to vary more globally all the thresholds.

3. Results & Discussion

We have submitted two runs on Ohsumed collection. The first one considers higher thresholds than the second one (the constant used in the product with the change scale is larger). Its utility score is positive (+1.1). We submitted it for comparison on utility criteria. The comparison is favourable since about 60% of the scores are above the median (table1).

	below-median	at-median	above-median
relief1	12	14	37

Table 1 : Comparison on utility criteria of adaptive filtering run.

BEST COPY AVAILABLE

We considered also smaller thresholds (decreasing the constant) for a second run in order to increase the number of selected documents which was too small for optimizing precision since when less than 50 document are selected a penalty is applied. This time, the utility score is approximately -1 and the corrected precision is approximately 0.17 (0.28 if not corrected). The comparison of our result with other systems optimized for precision is not favourable (table 2). However, it is to be noted that our system is not tuned to optimized the precision but utility.We think that the results could be improved if we set lower thresholds in order to keep more documents and then to avoid the under-50 penalty.

	below-median	at-median	above-median
reliefs2	42	11	10

Table 2 : Comparison on precision criteria of adaptive filtering run.

Globally, these very first results are encouraging, in particular for utility. They show that using a small number of words (20) to represent documents can perform as well as traditional information filtering systems in which much more words are considered. However, it is also necessary to consider word conjunctions.

4. Conclusion

In our information filtering approach, we take into account two implications, D->Q and Q->D. Morever, we developped a solution in order to take into account word conjunctions. Further experiments will be done to evaluate more precisely the avantages of each of these aspects.

Acknowledgment

This research has been funded by INRIA (Institut National de Recherche en Informatique et Automatique).

References

Brouard, C.(2000). *Construction et exploitation de réseaux Sémantiques Flous pour l'Extraction d'Information Pertinente : Le système RELIEFS.* Thèse de l'université Paris 6.

Nie, J. Y. (1988). *An outline of a general model for information retrieval.* Proceedings of the 11th Annual ACM Conference on Research and Development in Information Retrieval, Grenoble.

van Rijsbergen, C. J. (1986). A non-classical logic for information retrieval. *The Computer Journal, 29(6)*, 481-485.

Report on the TREC-9 Experiment: Link-Based Retrieval and Distributed Collections

Jacques Savoy, Yves Rasolofo

Institut interfacultaire d'informatique
Université de Neuchâtel (Switzerland)
E-mail: {Jacques.Savoy, Yves.Rasolofo}@unine.ch
Web page: http://www.unine.ch/info/

Summary

The web and its search engines have resulted in a new paradigm, generating new challenges for the IR community which are in turn attracting a growing interest from around the world. The decision by NIST to build a new and larger test collection based on web pages represents a very attractive initiative. This motivated us at TREC-9 to support and participate in the creation of this new corpus, to address the underlying problems of managing large text collections and to evaluate the retrieval effectiveness of hyperlinks.

In this paper, we will describe the results of our investigations, which demonstrate that simple raw score merging may show interesting retrieval performances while the hyperlinks used in different search strategies were not able to improve retrieval effectiveness.

Introduction

Due to the huge number of pages and links, browsing cannot be viewed as an adequate searching process, even with the introduction of tables of contents or other classifying lists (e.g., Yahoo!). As a result, effective query-based mechanisms for accessing information will always be needed. Search engines currently available on the web are not able to adequately access all available information [Lawrence 99], as they are inhibited by many drawbacks [Hawking 99].

In the first chapter, we will describe our experiments on the web track in which a large web text collection is divided into four sub-collections in order to keep inverted file size below the 2 GB limit. The second chapter will verify whether or not hyperlinks improved retrieval effectiveness based on four different link-based search models.

To evaluate our hypothesis, we used the SMART system as a test bed for implementing the OKAPI probabilistic model [Robertson 95]. This year our experiments were conducted on an Intel Pentium III/600 (memory: 1 GB, swap: 2 GB, disk: 6 x 35 GB) and all experiments were fully automated.

1. Distributed collections

To evaluate the retrieval effectiveness of various merging strategies, we formed four separate sub-collections (see Appendix 1). In this study, we assumed that each sub-collection used the same indexing schemes and retrieval procedures. A distributed context such as this more closely reflects local area networks or search engines available on the Internet than the meta search engines, where different search engines may collaborate to respond to a given user request [Le Calvé 00], [Selberg 99].

The following characteristics would more precisely identify our approach. A query was sent to all four text databases (no selection procedure were applied) and according to the four ranked lists of items produced, our search system merged them into a single result list presented to the user (Section 1.2). Before we describe the collection merging approaches, Section 1.1 will identify retrieval effectiveness measures achieved by various search models with the whole collection and with each of our four sub-collections.

1.1. Performance of sub-collections

From the original web pages, we retained only the following logical sections: <TITLE>, <H1>, <CENTER>, <BIG>, with the most common tags <P> (or <p>, together with </P>, </p>) being removed. Text delimited by the tags <DOCHDR>, </DOCHDR> were also removed. For long requests, various insignificant keywords were also removed (such as "Pertinent documents should include ..."). Moreover, search keywords appearing in the Title part of the topics were considered to have a term frequency of 3 (this feature has no impact on short requests).

For the web track, we conducted different experiments using the OKAPI probabilistic model in which the weight w_{ij} assigned to a given term t_j in a document D_i was computed according to the following formula:

$$w_{ij} = \frac{(k_1 + 1) \cdot tf_{ij}}{K + tf_{ij}}$$

$$\text{with } K = k_1 \cdot \left[(1 - b) + b \cdot \frac{l_i}{avdl}\right]$$

where tf_{ij} indicates the within-document term frequency, and b, k_1 are parameters. K represents the ratio between the length of D_i measured by l_i (sum of tf_{ij}) and the collection mean denoted by advl.

To index a request, the following formula was used:

$$w_{qj} = \frac{tf_{qj}}{k_3 + tf_{qj}} \cdot \ln[(N - df_j) / df_j]$$

where tf_{qj} indicates the search term frequency, df_j the collection-wide term frequency, N the number of documents in the collection, and k_3 is a parameter.

To adjust the underlying parameters of the OKAPI search model, we used advl = 900, b = 0.7625, k_1 = 1.5, and k_3 = 1000. These parameter values were set according to the best performance achieved on the WT2g (TREC-8). A slightly different parameter setting was suggested by Walker *et al.* [98] whereby advl = 900, b = 0.75, k_1 = 1.2, and k_3 = 1000. When using our parameter values, the corresponding label will be "OKAPI" while the second setting will be identified by adding an "R".

Two different query formulations were considered: (1) using only the Title section (T), or (2) all three logical sections (Title, Descriptive and Narrative, noted T-D-N). The data in Table 1 shows that retrieval effectiveness is significantly enhanced when topics include more search terms.

In order to build a single collection, we selected the first 500 retrieved items of 13 search strategies (corresponding to OKAPI and different vector-space approaches) and we added all relevant documents not retrieved by our various search models.

Table 1 provides a summary of the results of our various experiments. In this case, we reported the non-interpolated average precision at eleven recall values, based on 1,000 retrieved items per request. From this data we can see that the parameter setting used by Walker's *et al.* results in better performance (e.g., in the WEB9.1 sub-collection, the average precision increases from 19.47 to 20.30 (+4.3%)).

It is recognized that pseudo-relevance feedback (blind expansion) is a useful technique for enhancing retrieval effectiveness. In this study, we evaluated the OKAPI search model with and without query expansion in order to verify whether or not this technique might improve retrieval performance when faced with different query formulations.

In this study, we adopted Rocchio's approach [Buckley 96] where the parameter settings were chosen according to experiments done with the WT2g from the TREC data (TREC-8).

For a short request the values α=0.75, β=0.25 were assigned and the system was allowed to add to the original query those 50 search terms extracted from the 12-best ranked documents. For long queries, the parameters were set as follows: α=0.7, β=0.3 and the search engine was allowed to add to the original query those 40 search terms extracted from the 15 best-ranked documents. The resulting retrieval effectiveness is depicted in Table 1 under the label "XQ".

After examining sub-collections WEB9.1 and WEB9.3, there was some improvement in results, as depicted in Table 1. For example, based on our parameter setting and examining the WEB9.1 sub-collection, the average precision increased from 19.47 (label "OKAPI") to 21.44 (label "OKAPIXQ") (+10.1%). However, for the other two sub-collections, the average precision decreased (e.g. in WEB9.4, the average precision decreases from 19.26 to 18.24 (-5.3%)).

1.2. Merging procedure

Recent works have suggested solutions in which answer lists were merged in order to produce a unique ranked list of retrieved records. As a first approach, we might assume that each sub-collection contains approximately the same number of pertinent items and that the distribution of the relevant documents is the same across the answer lists. Based only on a ranking of the retrieved records, we might interleave the results in a round-robin fashion. According to previous studies [Voorhees 95], [Callan 95], the retrieval effectiveness of such interleaving schemes is around 40% below the performance achieved by a single retrieval scheme technique, with a single huge collection representing the entire set of documents. The third column of Table 2 confirms this finding but to a lesser extent (around -26.1% when dealing with short queries or -17.0% when examining Title, Descriptive and Narrative fields in the topics).

	Average Precision				
Query Title only Model	WEB9.1 46 queries 749 rel	WEB9.2 44 queries 600 rel	WEB9.3 43 queries 608 rel	WEB9.4 46 queries 660 rel	WEB9 50 queries 2,617 rel.
Okapi	19.47	20.85	16.09	19.26	19.55
OkapiR	20.30	21.32	16.52	19.51	19.86
OkapiXQ	21.44	20.89	17.73	18.24	19.43
OkapiNRXQ	21.70	20.67	18.98	18.33	19.31
Query T-D-N					
Okapi	32.61	30.26	28.09	28.44	27.25
OkapiNR	33.25	30.19	29.01	28.49	27.52
OkapiXQ					28.10
OkapiNRXQ	34.41	28.25	31.18	26.69	28.30

Table 1: Average precision of isolated sub-collections and the whole test collection

In order to account for the score achieved by the retrieved document, we might formulate the hypothesis that each sub-collection is managed by the same search strategy and that the similarity values are therefore directly comparable [Kwok 95]. Such a strategy, called raw-score merging, produces a final list, sorted by the retrieval status value computed by each separate search engine.

However, as demonstrated by Dumais [94], collection-dependent statistics in document or query weights may vary widely among sub-collections; and therefore, this phenomenon may invalidate the raw-score merging hypothesis.

The fourth column of Table 2 indicates the retrieval effectiveness of such merging approaches, depicting a relatively interesting performances in our case (degradation of around -5.3% for long requests or -14.9% for short queries). Thus, the raw-score merging seems to be a simple and valid approach when a huge collection is distributed across a local-area network and operating within the same retrieval scheme.

As a third merging strategy, we may normalize each sub-collection's similarity value (SIM(D_i, Q)) by dividing it by the maximum value in each result list. The fifth column in Table 2 shows its average precision, depicting surprisingly poor retrieval effectiveness (average reduction of -19.6% for short queries and -16.2% for long requests).

As a fourth merging strategy, Callan *et al.* [95] suggest using the CORI approach, which will first compute a score s_i for each sub-collection as follows:

$$\text{score}(t_j \mid db_i) = \text{defB} + (1\text{-defB}) \cdot \frac{df_i}{df_i + K} \cdot \frac{\log\left[\frac{db+0.5}{cf_j}\right]}{\log(db+1)} \quad \text{with } K = k \cdot \left[(1-b) + b \cdot \frac{ldb_i}{avldb}\right]$$

where t_j indicates a search keyword, db_i the ith collection, df_i the number of documents in the ith collection containing term t_j, cf_j the number of collections containing term t_j, db the total (number of collections equals to four in our case), ldb_i the number of indexing terms included in the ith corpus, avldb the mean value of ldb_i, where defB, b and k are three parameters. Xu & Callan [98] suggest assigning values to these constants (defB=0.4, k=200, and b=0.75, values used in this study). The previous equation is defined for one search term, and the score for a given collection is simply the sum over all keywords included in the current request.

The sub-collection score (noted s_i) is the first component used to merge the retrieved items. To obtain the score of a given retrieved item of the ith collection, the similarity between the request and the document is multiplied by a coefficient w_i computed as follows:

$$w_i = 1 + dbs \cdot [(s_i - S_m) / S_m]$$

	Average Precision (% change)				
Query Title Model	50 queries 2617 rel one coll	merge 50 queries 2617 rel round	merge 50 queries 2617 rel raw-score	merge 50 queries 2617 rel norm. score	merge 50 queries 2617 rel CORI
Okapi	19.55	13.88 (-29.0%)	17.59 (-10.0%)	15.94 (-18.5%)	15.83 (-19.0%)
OkapiR	19.86	14.44 (-27.3%)	17.81 (-10.3%)	16.37 (-17.6%)	15.99 (-19.5%)
OkapiXQ	19.43	14.54 (-25.2%)	15.96 (-17.9%)	15.07 (-22.4%)	15.31 (-21.2%)
OkapiNRXQ	19.31	14.89 (-22.9%)	15.87 (-17.8%)	15.44 (-20.0%)	15.28 (-20.9%)
		-26.1%	-14.9%	-19.6%	-20.2%
Query T-D-N					
Okapi	27.25	22.82 (-16.3%)	26.56 (-2.5%)	23.39 (-14.2%)	26.81 (-1.6%)
OkapiNR	27.52	23.19 (-15.7%)	26.75 (-2.8%)	23.94 (-13.0%)	26.87 (-2.4%)
OkapiXQ	28.10	23.09 (-17.8%)	25.99 (-7.5%)	23.28 (-17.2%)	25.94 (-7.7%)
OkaNRXQ	28.30	23.22 (-18.0%)	25.93 (-8.4%)	22.57 (-20.2%)	25.84 (-8.7%)
		-17.0%	-5.3%	-16.2%	-5.1%

Table 2: Average precision of different merging procedures

where dbs indicates the number of the selected collections (all in our case), s_i the score achieved by the ith collection and S_m the mean score over all collections. According to our evaluation, the mean average precision results in a degradation of around 20.2% for short queries and 5.1% for long requests. It is interesting to note that both the raw-score merging and the CORI approach result in good performances when dealing with long requests yet a decrease in performance when using short requests.

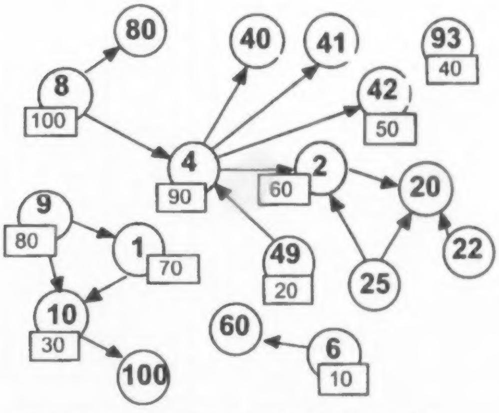


Figure 1: Starting situation for our link-based approaches

2. Link-based retrieval

Various retrieval strategies have been suggested in order to take account of hyperlinks, based on the assumption that links between documents indicate useful semantic relationships between related web pages [Kleinberg 98], [Brin 98], [Chakrabarti 99]. For example, Chakrabarti et al. [99] stated:

> "Citations signify deliberate judgment by the page author. Although some fractions of citations are noisy, most citations are to semantically related material. Thus the relevance of a page is a reasonable indicator of the relevance of its neighbors, although the reliability of this rule falls off rapidly with increasing radius on average." [Chakrabarti 99, p. 550-551]

With small variations, similar hypotheses are also cited by other authors [Kleinberg 98]. In order to verify the retrieval effectiveness of such assumptions, we have evaluated four different search strategies, namely our spreading activation approach in Section 2.1, our PAS search model in Section 2.2, Kleinberg's algorithm in Section 2.3 and the PageRank approach in Section 2.4. These search strategies will be described briefly using a small example.

As a first step, the search strategy computes the similarity between the given query and the documents, with values noted as SIM(D_i, Q). These values are depicted inside a rectangle in Figure 1. In this case, we can see that the first five retrieved

BEST COPY AVAILABLE

documents are D_8, D_4, D_9, D_1 and D_2. At this point various retrieval schemes will take note of the hyperlinks so that the retrieval effectiveness might hopefully be improved.

2.1. Spreading activation

In a first link-based strategy, we chose the spreading activation (SA) approach [Crestani 00]. In that method, the degree of match between a web page D_i and a query, as initially computed by the IR system (denoted $SIM(D_i, Q)$), is propagated to the linked documents through a certain number of cycles using a propagation factor. We used a simplified version with only one cycle and a fixed propagation factor λ for all links. In that case, the final retrieval status value of a document D_i linked to m documents is computed according to the following equation:

$$RSV(D_i) = SIM(D_i, Q) + \lambda \cdot \sum_{j=1}^{k} SIM(D_j, Q)$$

Using all the incoming and outgoing links, and for different values of the parameter λ, in most cases did not result in retrieval improvement within the WT2g corpus [Savoy 01]. In order to be more selective in the spreading phase, we only consider in this study the best outgoing and the best incoming link for each of the k best-ranked documents (the constant k was fixed to 15 in this paper and the parameter λ to 0.05). But, what do we mean by the best link?

Instead of considering the m web pages linked to a given document, we only consider the incoming link coming from the best ranked document. For the outgoing links, we adopt a similar point of view, taking into account only the link starting from the given document to the best rank web page.

For example, based on Figure 1, we do not follow all outgoing from D_4 but we activate only the hyperlink to D_2 (the rank of this document is better than for the others). Similarly, the best incoming link is the link between D_8 to D_4. Fixing the parameter λ to 0.1 and k to 5, the final retrieval status value of D_4, noted $RSV(D_4)$, will be :

$$RSV(D_4) = SIM(D_4, Q) + \lambda \cdot SIM(D_2, Q) + \lambda \cdot SIM(D_8, Q) =$$
$$90 + 0.1 \cdot 60 + 0.1 \cdot 100 = 106$$

The similarity value of non-retrieved documents (e.g., D_{20} in our example) will be set according the similarity achieved by the last retrieved item (10 in our example, 1,000 in the evaluation). The evaluation of other web pages included in our example is given in Table 3.

2.2. Probabilistic argumentation system

In a second set of experiments, we used our probabilistic argumentation systems (PAS) [Picard 98], in which we used a simplified version of our approach, whereby the final retrieval status value of a document (or its degree of support, denoted $DSP(D_i)$) might only be affected by its direct neighbors. In this case we do not need to keep track of inferences, and can derive a simple formula which might be considered to be a more refined spreading activation method. Instead of propagating a document's similarity value, we propagated its probability of being relevant.

In this approach, we must therefore first compute the relevance probability of a document D_i. To achieve this, we suggest using logistic regression methodology [Bookstein 92] and the natural logarithm of its rank as an explanatory variable. Such a computation will be noted $p(D_i \mid rank)$ [Le Calvé 00] and in accordance with the following formula:

$$P\left[D_i \mid rank\right] = \frac{e^{\alpha+\beta \cdot \ln(rank)}}{1+e^{\alpha+\beta \cdot \ln(rank)}}$$

in which α et β are parameters set to 0.7 and -0.8 respectively.

In a second step, this probability of relevance will be modified according to the neighbors of a given document. The individual contribution of a linked document D_j to D_i is given by $[p(D_j \mid rank) \cdot p(link)]$, instead of the $[SIM(D_j, Q) \cdot \lambda]$ used with the spreading activation technique.

Just as with the spreading activation experiments, using all incoming or outgoing links did not demonstrate any improvement, except in some cases when using the WT2g test collection [Savoy 01]. We then decided to include only the most important sources of evidence, the same way as for spreading activation. For example, we considered the initial rank of document D_i, the best incoming document D_{in} and the best outgoing document D_{out}.

This link-based retrieval approach will thus multiply the probability of linked document relevance by the probability of the link, denoted

$p(link_{in})$ for incoming hyperlinks or $p(link_{out})$ for outgoing links. The final degree of support corresponding to document D_i is computed as follow:

$$DSP(D_i) = 1 - (1 - p(D_i \mid rank)) \cdot [1 - p(D_{in} \mid rank) \cdot p(link_{in})] \cdot [1 - p(D_{out} \mid rank) \cdot p(link_{out})]$$

Fixing $p(link_{in})$=0.1 and $p(link_{out})$=0.2, and based on the situation depicted in Figure 1, computation of degree of support for Document 1 as follows:

$$\begin{aligned} DSP(D_1) &= 1 - (1 - p(D_1 \mid rank)) \cdot [1 - p(D_9 \mid rank) \cdot p(link_{in})] \cdot [1 - p(D_{10} \mid rank) \cdot p(link_{out})] = \\ & 1 - (1 - 0.3991) \cdot [1 - 0.4554 \cdot 0.1] \cdot [1 - 0.2762 \cdot 0.2] = \\ & 1 - (0.6009) \cdot [0.95446] \cdot [0.94476] = \\ & 1 - 0.5418 = 0.4582 \end{aligned}$$

Table 4 lists other results pertaining to the best ten retrieved items of Figure 1. For the results based on the web test collection, link probabilities are fixed as $p(link_{in})$ = 0.062, $p(link_{out})$ = 0.051, probability estimates are defined in [Savoy 01]. Finally, documents not belonging to the top 1000 have a similarity value equal to the similarity value obtained for the 1000th retrieved item.

2.3. Kleinberg's algorithm

As a third link-based approach, we have applied Kleinberg's algorithm [Kleinberg 98]. In this scheme, a web page pointing to many other information sources must be viewed as a "good" hub while a document with many web pages pointing to it is a "good" authority. Likewise, a document that points to many "good" authorities is an even better hub while a web page pointed to by many "good" hubs is an even better authority.

For document D_i after c+1 iterations, the updated formulas for the hub and authority scores $H^{c+1}(D_i)$ and $A^{c+1}(D_i)$ are:

$$A^{c+1}(D_i) = \sum_{D_j = parent(D_i)} H^c(D_j)$$

$$H^{c+1}(D_i) = \sum_{D_j = child(D_i)} A^c(D_j)$$

Rank	D_i	$SIM(D_i, Q)$	D_i	$RSV(D_i)$
1	8	100	8	109
2	4	90	4	106
3	9	80	9	87
4	1	70	1	78
5	2	60	2	69
6	42	50	42	50
7	93	40	93	40
8	10	30	10	37
9	49	20	49	20
10	6	10	20	16
			6	10

Table 3: Retrieval status value obtained by the spreading activation

Rank	D_i	$SIM(D_i,Q)$	$p(D_i \mid rank)$	D_i	$DSP(D_i)$
1	8	100	0.6682	8	0.7038
2	4	90	0.5363	4	0.5982
3	9	80	0.4554	9	0.4989
4	1	70	0.3991	1	0.4582
5	2	60	0.3572	2	0.4211
6	42	50	0.3244	42	0.3244
7	93	40	0.2980	10	0.3051
8	10	30	0.2762	93	0.2980
9	49	20	0.2577	20	0.2690
10	6	10	0.2419	49	0.2577
11			0.2419	6	0.2419

Table 4: Computation of the degree of support of our PAS search model

which is computed for the k best-ranked documents (defined as the root set) retrieved by a classical search model, togethcr with their children and parents (which defined the base set). The hub and authority scores were updated for five iterations (while the ranking did not change after this point), and a normalization procedure (dividing each score by the sum of all square values) was applied after each step.

As an example, we will refer to the initial situation shown in Figure 1. We fixed k = 5 and our root set was $\{D_8, D_4, D_9, D_1, D_2\}$, leading to the following base set $\{D_8, D_4, D_9, D_1, D_2, D_{80}, D_{40}, D_{41}, D_{42}, D_{20}, D_{25}, D_{49}, D_{10}\}$. Initially, the hub and authority score for each document is set to 1. In the first iteration, the hub score for D_4 corresponds to the sum of the authority values for its children (D_{40}, D_{41}, D_{42}, D_2) while its authority score is the sum of the hub scores of its parents (D_8, D_{49}). For other items belonging to the basic set, computation of these scores is depicted in Table 5.

After five iterations and using the normalization procedure, we obtained the ranked list depicted in Table 6. Taking the five best-ranked documents obtained by the traditional search engine into account and the top five documents retrieved according to the authority scores, we note that the intersection included only one item, namely D_2.

2.4. PageRank algorithm

Brin & Page [98] suggest a link-based search model called PageRank that first evaluated the importance of each web page based on its citation pattern. As for the spreading activation approach, the PageRank algorithm reranked the retrieved pages of a traditional search schemes according to the PageRank values assigned to the retrieved items.

In this approach, a web page will have a higher score if many web pages point to it. This value increases if there are highly scoring documents pointing to it. The PageRank value of a given web page D_i, value noted as $PR(D_i)$, having D_1, D_2, … D_m pages pointing to D_i, is computed according to the following formula:

$$PR(D_i) = (1 - d) + d \cdot [(PR(D_1) / C(D_1)) + \ldots + (PR(D_m) / C(D_m))]$$

where d is a parameter (set to 0.85 as suggested by [Brin 98] and $C(D_j)$ are the number of outgoing links for web page D_j.

The computation of the PageRank value can be done using an iterative procedure (five iterations were computed in our case). After each iteration, each PageRank value was divided by the sum of all PageRank values. Finally, as initial values, $PR(D_i)$ were set to 1/N where N indicates the number of documents in the collection.

Based on our example, the result list achieved by using the PageRank algorithm is depicted in Table 8.

2.5. Evaluation

The retrieval effectiveness of the four link-based search model are shown in Table 9. From this table, it seems clear that links do not seem an appropriate source of information about document contents, and they seem to provide less information than do the bibliographic references or co-citation schemes used in our previous studies [Savoy 96]. The poor results depicted by Kleinberg's approach or PageRank algorithm raise some questions: Is our implementation without bugs? Can other teams confirm these findings? Have the underlying parameters the good values?

Our official runs were produced using the raw-score merging, where three were based only on the Title portion of the requests (NEtm, NENRtm, NENRtmLpas) and three were based on all logical sections of the queries (NEnm, NEnmLpas, NEnmLsa). Three of them were link-based retrievals (ending by Lpas or Lsa indicating the PAS or spreading activation approach).

For the two types of requests, our official runs included a spelling check performed automatically by the Smalltalk-80 system. This feature has a positive effect for short queries (e.g., 15.96 vs. 17.54 (+9.9%)) but not for long ones (25.99 vs. 24.99 (-3.8%)).

Conclusion

The various experiments carried out within the web track demonstrated that:

- Hyperlinks do not result in any significant improvement (at least as implemented in this study). Link information seems to be marginally useful when the retrieval system produces relatively high retrieval effectiveness;

- Pseudo-relevant feedback techniques (blind query expansion) usually result in significant improvement but setting the underlying parameters based on another test collection may lead to a decrease in retrieval effectiveness;
- Longer topic descriptions (Title, Description and Narrative) improve the retrieval performance significantly over short queries built only from the Title section;
- It seems that the raw-score approach might be a valid first attempt for merging result lists provided by the same retrieval model.

Acknowledgments

The authors would like to thank C. Buckley from SabIR for allowing us the opportunity to use the SMART system. This research was supported by the SNSF (Swiss National Science Foundation) under grant 21-58'813.99.

D_i	$H^0(D_i)$	Author comput	$A^1(D_i)$	$A^0(D_i)$	Hub comput	$H^1(D_i)$
8	1		0	1	1 + 1	2
4	1	1 + 1	2	1	1 + 1 + 1 + 1	4
9	1		0	1	1 + 1	2
1	1	1	1	1	1	1
2	1	1 + 1	2	1	1	1
40	1	1	1	1		0
41	1	1	1	1		0
42	1	1	1	1		0
49	1		0	1	1	1
80	1	1	1	1		0
20	1	1 + 1	2	1		0
25	1		0	1	1 + 1	2
10	1	1 + 1	2	1		0

Table 5: Computation of the hub and authority scores for our example

Rank	D_i	$SIM(D_i, Q)$	D_i	$A^5(D_i)$	D_i	$H^5(D_i)$
1	8	100	2	0.1239	4	0.1501
2	4	90	42	0.0762	25	0.0723
3	9	80	41	0.0762	9	0.0241
4	1	70	40	0.0762	8	0.0241
5	2	60	20	0.0667	2	0.0222
6	42	50	4	0.0413	49	0.0148
7	93	40	10	0.0413	1	0.0148
8	10	30	80	0.0254	80	0
9	49	20	1	0.0254	42	0
10	6	10	9	0	41	0

Table 6: Computation of the hub and authority scores after five iterations

Rank	D_i	$SIM(D_i, Q)$	Rank	D_i	$PR(D_i)$
1	8	100	1	10	0.2710
2	4	90	2	4	0.2548
3	9	80	3	2	0.2146
4	1	70	4	1	0.1849
5	2	60	5	42	0.1797
6	42	50	6	93	0.15
7	93	40	7	49	0.15
8	10	30	8	9	0.15
9	49	20	9	8	0.15
10	6	10	10	6	0.15

Table 8: Ranked list obtained in our example by the traditional and the PageRank approach

	without normalizat.		with normalization	
D_i	$PR^1(D_i)$	$PR^5(D_i)$	$PR^1(D_i)$	$PR^5(D_i)$
1	0.1736	0.2138	0.1925	0.1849
2	0.1854	0.2863	0.2138	0.2146
4	0.2208	0.3413	0.2775	0.2548
6	0.15	0.15	0.15	0.15
8	0.15	0.15	0.15	0.15
9	0.15	0.15	0.15	0.15
10	0.2208	0.3954	0.2775	0.2710
20	0.2681	0.5846	0.3625	0.3547
22	0.15	0.15	0.15	0.15
25	0.15	0.15	0.15	0.15
40	0.1618	0.2225	0.1713	0.1797
41	0.1618	0.2225	0.1713	0.1797
42	0.1618	0.2225	0.1713	0.1797
49	0.15	0.15	0.15	0.15
60	0.1972	0.2775	0.235	0.2198
80	0.1736	0.2138	0.1925	0.1849
93	0.15	0.15	0.15	0.15
100	0.1972	0.4861	0.235	0.2762

Table 7: Computation of the PageRank values with and without normalization

	Average Precision				
Query Title Model	merge raw-score	SA	PAS	Kleinberg	PageRank
Okapi	17.59	14.64	17.57	0.18	2.82
OkapiR	17.81	14.59	17.76	0.19	2.79
OkapiXQ	15.96	13.43	15.91	0.17	2.37
OkapiNRXQ	15.87	13.48	15.85	0.17	2.69
Query T-D-N					
Okapi	26.56	23.80	26.43	0.36	3.09
OkapiNR	26.75	24.10	26.65	0.25	3.14
OkapiXQ	25.99	22.27	25.87	0.31	3.11
OkaNRXQ	25.93	22.57	25.82	0.25	3.13

Table 9: Average precision of different link-based approaches

Official run name	Corresponding run name	Average Pre.	# ≥ Median	# Best
NEtm	OKAPIXQ	17.54	41	3
NENRtm	OKAPIRXQ	17.43	41	2
NENRtmLpas	OKAPIRXQ + PAS	17.36	40	1
NEnm	OKAPIXQ	24.99	45	4
NEnmLpas	OKAPIRXQ + PAS	24.88	43	0
NEnmLsa	OKAPIRXQ + SA	21.85	41	0

Table 10: Summary of our official runs for the web track

References

[Bookstein 92] A. Bookstein, E. O'Neil, M. Dillon, D. Stephens: Applications of loglinear models for informetric phenomena. Information Processing & Management, 28(1), 1992, 75-88.

[Brin 98] S. Brin, L. Page: The anatomy of a large-scale hypertextual web search engine. WWW8, 1998, 107-117.

[Buckley 96] C. Buckley, A. Singhal, M. Mitra, G. Salton: New retrieval approaches using SMART. TREC-4, 1996, 25-48.

[Callan 95] J. P. Callan, Z. Lu, W. B. Croft: Searching distributed collections with inference networks. ACM-SIGIR'95, 21-28.

[Chakrabarti 99] S. Chakrabarti, M. Van den Berg, B. Dom: Focused Crawling: A new approach to topic-specific web resource discovery. WWW8, 1999, 545-562.

[Crestani 00] F. Crestani, P. L. Lee: Searching the web by constrained spreading activation. Information Processing & Management, 36(4), 2000, 585-605.

[Dumais 94] S. T. Dumais: Latent semantic indexing (LSI) and TREC-2. TREC'2, 1994, 105-115.

[Hawking 99] D. Hawking, N. Craswell, P. Thistlewaite, D. Harman: Results and challenges in web search evaluation. WWW8, 1999, 243-252.

[Kleinberg 98] J. Kleinberg: Authoritative sources in a hyperlinked environment. Proceedings of 9th ACM-SIAM Symposium on Discrete Algorithms, 1998, 668-677.

[Kwok 95] K. L. Kwok, L. Grunfeld, D. D. Lewis: TREC-3 ad-hoc, routing retrieval and thresholding experiments using PIRCS. TREC-3, 1995, 247-255.

[Lawrence 99] S. Lawrence, C. Lee Giles: Accessibility of information on the web. Nature 400 (6740), 1999, 107-110.

[Le Calvé 00] A. Le Calvé, J. Savoy: Database merging strategy based on logistic regression. Information Processing & Management, 36(3), 2000, 341-359.

[Picard 98] J. Picard: Modeling and combining evidence provided by document relationships using PAS systems. ACM-SIGIR'98, 182-189.

[Robertson 95] S. E. Robertson, S. Walker, M. M. Hancock-Beaulieu: Large test collection experiments on an operational, interactive system: OKAPI at TREC. Information Processing & Management, 31(3), 1995, 345-360.

[Savoy 96] J. Savoy: Citation schemes in hypertext information retrieval. In Information retrieval and hypertext, M. Agosti, A. Smeaton (Eds), Kluwer, 1996, 99-120.

[Savoy 01] J. Savoy, J. Picard: Retrieval effectiveness on the web. Information Processing & Management, 2001, to appear.

[Selberg 99] E. W. Selberg: Towards comprehensive web search. Ph.D. Thesis, University of Washington, 1999.

[Voorhees 95] E. M. Voorhees, N. K. Gupta, B. Johnson-Laird: Learning collection fusion strategies. ACM-SIGIR'95, 172-179.

[Walker 98] S. Walker, S. E. Robertson, M. Boughamen, G. J. F. Jones, K. Sparck Jones: Okapi at TREC-6: Automatic ad hoc, VLC, routing, filtering and QSDR. TREC-6, 1998, 125-136.

[Xu 98] J. Xu, J. P. Callan: Effective retrieval with distributed collections. ACM-SIGIR'98, 112-120.

Appendix 1: Statistics describing our various sub-collections

Collection	WEB9.1	WEB9.2	WEB9.3	WEB9.4	WEB9
Size (in MB)	2,799 MB	2,754 MB	2,790 MB	2,690 MB	11,032 MB
# of documents	414,914	423,965	442,711	410,506	1,692,096
# of relevant doc.	749	600	608	660	2617
# of queries	46	44	43	46	50
mean	16.2826	13.6364	14.1395	14.3478	52.34
standard error	25.0986	21.826	21.3637	22.1873	84.1405
maximum	157	105	133	124	519
for # query	(#q:495)	(#q:495)	(#q:495)	(#q:495)	(#q:495)
minimum	1	1	1	1	1
for # query	(#q:461)	(#q:461)	(#q:464)	(#q:456)	(#q:473)
size invert. file doc.nnn	674.2 MB	642.1 MB	655.6 MB	635.4 MB	
# indexing terms	3,428,795	2,827,067	3,607,359	3,537,393	
max df	189,386	207,892	228,922	191,208	
Indexing time (real)	1:05:17	1:00:28	1:00:18	1:00:49	

Table A.1: Some statistics about the four sub-collections of the Web corpora

TREC-9 Cross-Language Information Retrieval (English - Chinese) Overview

Fredric Gey and Aitao Chen
UC DATA and SIMS
University of California, Berkeley
e-mail: gey@ucdata.berkeley.edu,aitao@sims.berkeley.edu

Abstract

Sixteen groups participated in the TREC-9 cross-language information retrieval track which focussed on retrieving Chinese language documents in response to 25 English queries. A variety of CLIR approaches were tested and a rich set of experiments performed which measured the utility of various resources such as machine translation and parallel corpora, as well as pre- and post-translation query expansion using pseudo-relevance feedback.

1 Introduction

For TREC-9 the cross-language information retrieval task was to utilize English queries against Chinese documents. This aspect of multilingual information access at TREC-9 was the seventh year in which non-English document retrieval was tested and evaluated, and the fourth year for which cross-language information retrieval has been experimented with. In TREC-3, retrieval of 25 queries against a Mexican newspaper corpus was tested by four groups. Spanish language retrieval was evaluated in TREC-3, TREC-4 (another 25 queries for the same Mexican corpus), and TREC-5 (where an European Spanish corpus was used). In TREC-5 a Chinese language track was introduced using both newspaper (People's Daily) and newswire (XinHua) sources from People's Republic of China and 25 Chinese queries with an English translation supplied. The TREC-5 corpus was represented with the GB character set for the simplified Chinese language of PRC. Chinese monolingual experiments on this collection were done in TREC-5 and TREC-6 and sparked serious research into Chinese text segmentation methods using dictionary methods as well as statistical methods using measures such as mutual information. Comparisons have been made with simple overlapping bigram segmentation methods for monolingual Chinese retrieval. TREC conferences TREC-6, TREC-7 and TREC-8 has cross language tracks which focussed upon European languages (English, French, German, and later Italian). Following TREC-8 the venue for evaluating European language retrieval moved to Europe with the Cross-Language Evaluation Forum (CLEF) first held in Lisbon in September 2000 [9].

BEST COPY AVAILABLE

2 Task Description

As in past TREC cross-language information retrieval evaluations, the task for each group was to match topics in one language (English in this case) against documents in another language (Chinese) and return a ranked list of the top 1000 documents associated with each topic. Multiple runs were allowed for each group but one run using only the title and description field was required. Evaluation then proceeded by pooling ranks and manual examination of the pools by human judges who decide upon the relevance or irrelevance of each document in the pool. Once relevance judgments were established the usual measures of recall and precision could be computed upon the ranked list of each entry.

2.1 Topics

Twenty-five topics in English (numbers CH55-CH79) were created at NIST. Two typical topics are Topic 56 (human rights violations) and Topic 79 (livestock in China):

```
<top>
<num> Number: CH56
<title> human rights violations
<desc> Description:
What human rights violations have occurred in countries
outside of China according to the Chinese press.
<narr> Narrative:
Reports of human rights violations in China are not
relevant.
</top>

<top>
<num> Number: CH79
<title> livestock in China
<desc> Description:
What kinds of livestock are being raised in China?
<narr> Narrative:
A document that discusses livestock farming in China,
but is not specific about the kind of livestock is
not relevant.
</top>
```

These topics demonstrate two kinds of difficulty. For topic CH56, the limitation of relevant human rights violations to 'countries outside China' is one of discrimination between human rights news stories concerned within China and those whose focus is other than China. Topic CH79 illustrates the use of a general term (livestock) while requesting specificity (e.g. pigs) within the documents returned.

2.2 Documents

The corpus for TREC-9's CLIR evaluation consisted of 126,937 documents (188 megabytes in size) with newspaper sources from Hong Kong for the periods 1998-1999. In distinction from the earlier TREC

Chinese corpus, these sources were written in the richer traditional Chinese character set, encoded in the BIG5 encoding. In particular the source documents came from:

- Hong Kong Commercial Daily (Aug 11, 1998 - Jul 31, 1999)
- Hong Kong Daily News (Feb 1, 1999 - Jul 31, 1999)
- Ta kung pao (Oct 21, 1998 - Mar 4, 1999)

3 Participants and General Approach

Sixteen groups participated in the TREC-9 Chinese evaluation, listed here in alphabetical order:

BBN Technologies
Fudan University (PRC)
IBM T.J. Watson Research Center
Johns Hopkins University (Applied Physics Laboratory)
Korea Advanced Institute of Science and Technology
Microsoft Research, China
MNIS-TextWise Labs
National Taiwan University
Queens College, CUNY
RMIT University (Australia)
Telecordia Technologies, Inc.
The Chinese University of Hong Kong
Trans-EZ Inc.
University of California at Berkeley
University of Maryland
University of Massachusetts

The majority of approaches utilized word or phrase translation from English to Chinese by lookup in bilingual dictionaries or word lists. A number of groups used the Linguistic Data Consortium's English-Mandarin word list of approximately 120,000 pairs of words. Other dictionaries included the CETA (Chinese-English Translation Assistance) dictionary and the KingSoft online bilingual dictionary as well as local (proprietary) dictionaries.

Other approaches (in particular BBN) made use of statistical association models to create bilingual dictionaries from the alignment of parallel English-Chinese Corpora. Corpora used for development of resources or pre/post query expansion included:

- LDC parallel Hong Kong SAR Law
- LDC parallel Hong Kong SAR News

- Academica Sinica Balanced Corpus (ASBC)
- TREC-6 (People's Daily and Xinhua News Agency)
- Foreign Broadcast Information Service (FBIS) data
- Bilingual data harvested from the WWW
- Other local (proprietary) mono and bilingual corpora

In addition a few commercial machine translation software packages were used and coupled with other resources.

Extensive experimentation was done by some groups with query expansion, both before query translation from English to Chinese and after translation using blind feedback from the top ranked documents of an initial retrieval.

4 Experimental and methodological details by group

This section provides a summary of experiments run and methodological approaches tested by the eight groups with best-performing English-Chinese crosslingual runs, according to the official results (see Results section below). Experiments and approaches which seem unique are given more description in this section. Readers are directed to the individual papers for more detail.

4.1 BBN

BBN [12] extended the hidden Markov model (HMM) for monolingual retrieval to cross-language retrieval by incorporating into the model the word translation probabilities. Two manually created lexicons (i.e, the LDC wordlist and CETA) and two parallel corpora (i.e, the Hong Kong News and Hong Kong Law) were used to translate English query words into Chinese. The parallel texts were first aligned at the sentence level iteratively using WEAVER, a statistical machine translation toolkit developed at Carnegie Mellon University. Then WEAVER was applied to the sentence-aligned parallel texts to estimate word translation probabilities. When the translation resources were used individually, the Hong Kong News corpus yielded the best performance, probably because of similarity in topics covered in the test documents and the Hong Kong News corpus. However when all four translation resources were combined, the overall precision was substantially better. Unlike many participating groups of the cross-language track, BBN did not attempt phrasal translation and disambiguation of translation terms. A number of retrieval runs were performed to test the impact of query expansion for three levels of query length. The results showed over 10% improvement for overall precision for both pre-translation and post-translation query expansion when either one was applied alone. But when both pre- and post-translation expansion were applied, the post-translation expansion did not further improve the overall precision execept for the short queries consisting of only titles. In their official monolingual run, the Chinese text was segmented into words using the built-in segmentor in BBN's IdentiFinder.

BEST COPY AVAILABLE

The monolingual performance was slightly lower than the best cross-language retrieval performance. However later it was found that when the bigrams and unigrams were used in indexing, the monolingual performance increaded from .2888 to .3779.

4.2 Microsoft Research, China

Microsoft Research China group used a slightly modified version of SMART as their retrieval system [4]. A series of experiments were carried out to test the impact on retrieval performance using different indexing units and their combination. The results show that combining word-indexing and character-indexing works well for Chinese monolingual retrieval. They also used NLWin, a natural language processing system developed by Microsoft, to identify multi-word phrases and unknown words in the Chinese texts. They developed a co-occurrence based method to disambiguate translation terms, and a phrase detection and translation technique which improved the retrieval performance over the primitive dictionary-based translation. The phrases were identified using NLWin, and complex phrases were translated into Chinese based on a statistical model that maximizes the probability of phrase translation patterns and bigram probabilities estimated from a bigram language model trained on a large Chinese corpus. An interesting feature of the phrasal translation was that the Chinese words were put in the approriate order which may differ from the order of the source English words. About 125 MB of Chinese and English parallel texts were automatically mined from the Internet. A statistical translation model which is a variant of the IBM model was applied to the sentence-aligned parallel text to estimate word translation probabilities. When translation disambiguation and phrasal translations were augmented by statistical translation, the cross-language retrieval performance was as good as that obtained using IBM HomePage Dictionary 2000, a commercial Engish-Chinese machine translation system. The best cross-language retrieval run MSRCN2 combined bilingual lexicon, parallel texts mined from the Internet, and the machine translation system. Both pre- and post-translation query expansion were tried, however the pre-translation query expansion did not improve the overall precision.

4.3 Fudan University, China

The document scoring function used by Fudan University group was based on the maximum likelihood ratio formula developed by MIT. A number of rule-based named entity extractors were used to identify words that are not in the segmentation dictionary. In addition, the occurrence frequency and mutual information between characters of unidentified strings were used to identify unknown words. The translation resources used for query translation consist of three dictionaries: a general English-Chinese dictionary, a technical terminology dictionary, and an idiom dictionary. The translated queries were further expanded using a Chinese thesaurus of nearly 70,000 entries. The best cross-language run was the one without pseudo relevance feedback [11].

4.4 Chinese University of Hong Kong

The CUHK group translated the queries into Chinese by considering two adjacent words each time. Among all possible translation pairs found in a bilingual dictionary, the one with the hightest similarity was chosen as the final translation for the source adjacent words. They experimented with pre- and post-translation query expansion using Rocchio relevance feedback within the SMART retrieval system. The pre-translation query expansion improved the overall precision from .1862 to .2642, an increase of 42% over the baseline run. However the post-translation did not gain any further improvement [5].

BEST COPY AVAILABLE

4.5 Queens College, New York

The Queens College group used a commercial machine translation software named HuaJian and the LDC wordlist augmented with an additional 6,000 translation pairs extracted from the Hong Kong Laws corpus to translate the queries. For dictionary lookup, up to six translation terms were kept for each query term. An equal weight was assigned to each translation terms of the same source query term. The final result for a cross-language run was produced by combining the result using the MT-translated queries and the result using the dictionary-translated queries. The best cross-language run named HxD combined MT and augmented LDC wordlist translations with pre- and post-translation query expansion and Chinese collection enrichment [6].

4.6 University of Massachusetts

The University of Massachusetts group used their INQUERY system with Local Context Analysis (LCA) technique for query expansion. The Chinese queries were translated into English by looking up multi-word phrases or words in a bilingual dictionary built by merging two Chinese-English dictionaries with an English-Chinese dictionary. Multiple translations were retained and treated as synonyms. Both pre- and post-translation query expansion using LCA were tried. The post-translation query expansion gained very little improvement [1].

4.7 IBM Research

The IBM group used a character-based statistical model to translate the English queries into Chinese, and a word-based statistical model supplemented with the LDC dictionary to translate the Chinese documents into English, both models being trained on Hong Kong News and Hong Kong Law parallel corpora. Their official run was a merging of the results from three runs based statistical query translation, commercial MT-based query translation, and statistical document translation [3].

4.8 Korea Advanced Institute for Science and Technology (KAIST)

The KAIST group experimented with query translations using bilingual dictionaries and machine translation systems. The cross-language run using two bilingual dictionaries of 50,000 and 15,000 entries respectively outperformed the one using two machine translation systems. They observed that some of the proper names in the queries are spelled out in Chinese Pinyin (e.g., Daya Wan) and attempted to obtain the Chinese names based on a Chinese pinyin table and the occurrence statistics of the characters in the Chinese collection [7].

5 Relevance judgments and pool contributions

In order to create a pool of documents for each topic for human evaluation of relevance, each participating group was invited to nominate a single entry run from the monolingual and/or cross-lingual tasks to be included in the judgments. This produced 39 cross-lingual runs and 13 monolingual runs. All but one of the runs was automatic. The top 50 ranked documents were taken from each nominated run and added to the pool to be evaluated. As usual duplicated documents from runs with overlap are removed to produce a unique list of documents for each topic. The resulting document pools had mean size of 598 documents (39 percent of the maximum pool size) to be read by the judges. The relevant documents over the pools came from the component run-types as follows: thirteen percent were only found by monolingual runs, twenty-eight percent came from crosslingual runs only, while the remaining 59 percent were found in both monolingual and crosslingual runs. Figure 1 shows the number of unique documents contributed by site.

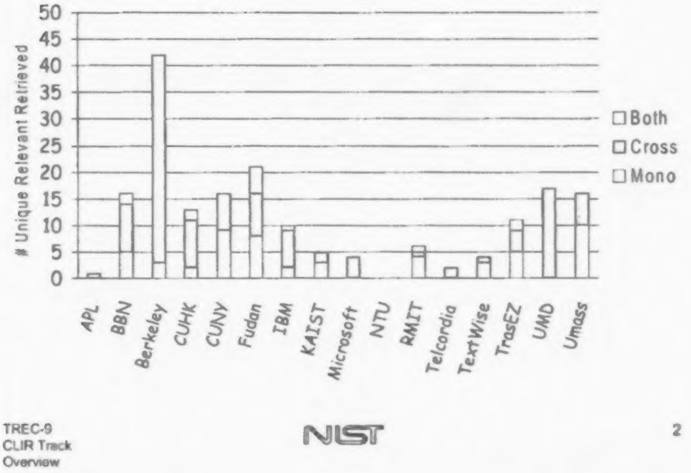


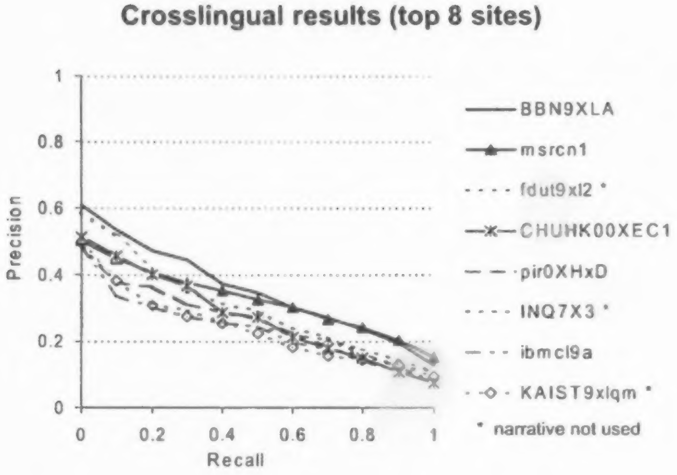
Figure 1: Unique Relevant Documents by Site

The only groups which contributed more than 20 relevant documents to the pool were Fudan University [11] and Berkeley [2].

6 Results

Figure 2 displays the recall-precision graph for the top eight best-performing crosslingual sites (the figure shows only the best run from each site). Although some groups seem to have clearly outperformed others, readers are cautioned that the evaluation only covered twenty-five queries, and it is unlikely that sufficient statistical signficance could be attained to confirm the rankings. The team from BBN outperformed all others with their hybrid combination of methods using a hidden markov ranking model, parallel corpora, and query expansion.

The reporting in the graph mixes run modes, since three runs used only title and description, while



2

Figure 2: TREC-9 Cross-Language Retrieval Results

the others used the narrative portion of the topic as well.

The monolingual results for the top eight sights are displayed in figure 3. While they show very little difference between sites, two sites, Johns Hopkins - apl9xmon [8] and TextWise - TWmono [10] clearly performed significantly better on their monolingual than crosslingual runs. On the other hand, the best precision at 0.0 recall of 0.7079 from Berkeley (BRKCCA1) exceeded the best official CLIR run of 0.6078 by BBN. Finally, the overall average precision of the best official monolingual run (apl9xmon - 0.3085) trails the best crosslingual run (bbn9xla - 0.3485). BBN noted this discrepancy and attributed it in part to lack of query expansion in other bigram-based methods (BBN's official monolingual run used word-based indexing). BBN later implemented a bigram/unigram based monolingual algorithm with query expansion and achieved an overall monolingual precision of 0.3779 [12].

7 Summary and Outlook

The TREC-9 crosslingual information retreival task focussed this year on English-Chinese retrieval. Major experiments were undertaken using combinations of machine-readable dictionaries, machine translation software, and parallel corpora of news stories, legal documents, and bilingual sites mined from the

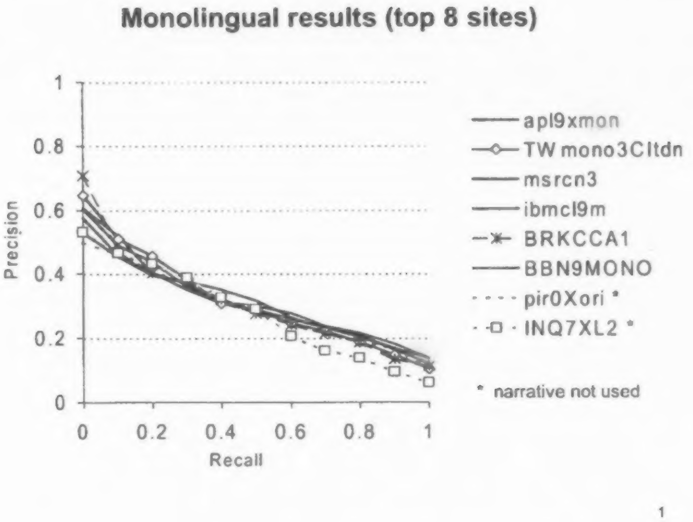


Figure 3: TREC-9 Monolingual Chinese Results

WWW. These were coupled with a variety of pre and post query expansion techniques. Many participating groups ran experiments which showed the contribution to overall precision of each component in the combination. The best performance was achieved by combining many of these techniques and by extensive use of the supportive resources.

In 2001 the TREC cross-language track will move from Chinese experiments to retrieving from a collection Arabic documents using either English or French queries. However, CLIR experiments with Chinese collections will continue with the NTCIR evaluation organized and hosted by the National Institute of Informatics of Japan (http://research.nii.ac.jp/ntcir/workshop/work-en.html).

We are grateful to Paul Over of NIST who supplied the basic information contained in our figures. His original Powerpoint presentation provided the basic outline from which this overview was written.

References

[1] J. Allan, M. Connell, W. B. Croft, F.-F. Feng, D. Fisher, , and X. Li. Inquery at trec-9. In D. K. Harman and E. Voorhees, editors, *in this volume*, 2001.

BEST COPY AVAILABLE

[2] A. Chen, H. Jiang, and F. Gey. English-chinese cross-language ir using bilingual dictionaries. In D. K. Harman and E. Voorhees, editors, *in this volume*, 2001.

[3] M. Franz, S. McCarly, and W.-J. Zhu. English-chinese information retrieval at ibm. In D. K. Harman and E. Voorhees, editors, *in this volume*, 2001.

[4] J. Gao, J.-Y. Nie, J. Zhang, E. Xun, Y. Su, M. Zhou, and C. Huang. Trec-9 clir experiments at msrcn. In D. K. Harman and E. Voorhees, editors, *in this volume*, 2001.

[5] H. Jin and K.-F. Wong. Trec-9 clir at cuhk, disabmiguation by similarity values between adjacent words. In D. K. Harman and E. Voorhees, editors, *in this volume*, 2001.

[6] K. Kwok, L. Grunfeld, N. Dinstl, and M. Chan. Trec-9 cross-language, web and question answering track experiments using pircsi. In D. K. Harman and E. Voorhees, editors, *in this volume*, 2001.

[7] K.-S. Lee, J.-H. Oh, J.-X. Huang, J.-H. Kim, and K.-S. Choi. Trec-9 experiments at kaist: Qa, clir and batch filtering. In D. K. Harman and E. Voorhees, editors, *in this volume*, 2001.

[8] P. McNamee, J. Mayfield, and C. Piatko. The haircut system at trec-9. In D. K. Harman and E. Voorhees, editors, *in this volume*, 2001.

[9] C. Peters, editor. Springer, Lecture Notes in Computer Science Series No. 2069, 2001.

[10] M. Ruiz, S. Rowe, M. Forrester, and P. Sheridan. Cindor trec-9 english chinese evaluation (mnis-textwise labs). In D. K. Harman and E. Voorhees, editors, *in this volume*, 2001.

[11] L. Wu, X. Huang, Y. Guo, B. Liu, and Y. Zhang. Fdu at trec-9: Clir, filtering and qa tasks. In D. K. Harman and E. Voorhees, editors, *in this volume*, 2001.

[12] J. Xu and R. Weischedel. Trec-9 cross-lingual retrieval at bbn. In D. K. Harman and E. Voorhees, editors, *in this volume*, 2001.

English-Chinese Cross-Language IR using Bilingual Dictionaries

Aitao Chen*, Hailing Jiang* and Fredric Gey†
*School of Information Management and Systems
†UC Data Archive & Technical Assistance (UC DATA)
University of California at Berkeley, CA 94720, USA
{aitao, hjiang1}@sims.berkeley.edu, gey@ucdata.berkeley.edu

Abstract

This report describes the English-Chinese cross-language retrieval experiments at Berkeley for TREC-9 Cross-Language Information Retrieval track. We present a simple and effective Chinese word segmentation method and compare the cross-language retrieval performance of two bilingual dictionaries for query translation.

1 Introduction

In TREC-9 we only participated in the English-Chinese cross-language information retrieval (CLIR) track. We performed one Chinese monolingual retrieval run and three English-Chinese cross-language retrieval runs. Our approach to the cross-language retrieval was to translate the English topics into Chinese by dictionary lookup. An English/Chinese bilingual wordlist compiled by Linguistic Data Consortium and an online English/Chinese bilingual dictionary were used in our cross-language retrieval experiments.

The four official runs we submitted are BRKCCA1, BRKECA1, BRKECA2, and BRKECM1. The BRKCCA1 is a monolingual run, the other three being English-Chinese cross-language runs. The BRKECA1 and BRKECA2 runs are automatic, while the BRKECM1 is manual.

For all four runs, the same document ranking algorithm based on logistic regression technique was used. The details on our ranking algorithm can be found in [2].

2 Word Segmentation

The documents and queries in most text retrieval systems are indexed by the words occurring in the text. For languages such as English in which words are separated by blank space, it is simple to index text by words. To index Chinese text by words, however, one first needs to identify words in the text since word boundaries are not explicitly marked in Chinese text. There is a large literature on Chinese word segmentation. We will not attempt to survey this field. Two recent papers on Chinese word segmentation are presented by Dai and Loh in [4] and Sun et al. in [9]. Both corpus-based statistical methods and dictionary-based methods have been developed to break a sentence into individual words. If one has a Chinese word dictionary, one could match the text against the dictionary and output as a word the longest sequence of characters that matches an dictionary entry. When a dictionary is not available, one could collect large amount of Chinese text and attempt to discover words by examing the occurrence patterns of the characters in the corpus. A major problem with dictionary-based word segmentation methods is the dictionary coverage. The corpus-based or statistical methods can be easily applied to a new collection of Chinese text since they do not use word dictionaries. The overlapping bigram indexing is simple, efficient and effective as well [7]. One problem with bigram indexing is that the indexing file produced is two to three times as big as the size of the raw text. Here we refer to single Chinese characters as unigrams and two-character Chinese terms as bigrams.

We present a method that is equally efficient and effective as bigram indexing, but produces a much smaller index file than the overlapping bigram indexing. Our method is similar to but less general than the work presented by Ge et al. in [5]. Our method breaks a sentence into unigrams and bigrams by maximizing the probability of the sentence. Here we assume that unigrams and bigrams occur independently in the corpus. For a segmented sentence $S = w_1 w_2 \ldots w_m$, if we assume words occur independently, then the probability of the sentence S can be expressed as follows:

$$
\begin{aligned}
P(S) &= P(w_1 w_2 \ldots w_m) && (1)\\
&= P(w_1)P(w_2)\ldots P(w_m) = \prod_{i=1}^{m} P(w_i) && (2)
\end{aligned}
$$

BEST COPY AVAILABLE

since we do not know how to break a sentence into words in advance, we will consider all possible ways of segmenting a sentence and estimate the probability of every segmentation given a sentence. We can then use the segmentation of the highest probability to break up the sentence into words. The number of possible ways to break a sentence of n characters into words is 2^{n-1} when a word can be arbitrarily long. However, when a word is limited to one or two characters, the number of possible ways to segment a sentence of n characters can be expressed by the recurrence relation $N(n) = N(n-1) + N(n-2)$, where $N(n)$ is the number of ways to break a sentence of n characters into one or two-character words and $N(0) = 0, N(1) = 1, N(2) = 2$. When a sentence is short, one can easily enumerate all possible ways of segmenting the sentence and compute their associated probabilities, then choose the segmentation of the highest probability. But when a sentence is long, the number of possible segmentations is exponential, it is no longer practical to enumerate all possible ways of breaking the sentence and estimate their probabilities. However one can apply dynamic programming technique to find out the most likely segmentation efficiently without computing the probabilities of all possible segmentations of a sentence. The best way of breaking a sentence of n characters can be recursively expressed as follows:

$$P(S_{1,n}) = MAX(P(S_{1,n-1})P(C_n), P(S_{1,n-2})P(C_{n-1}C_n))$$

where $S_{1,n} = C_1C_2 \dots C_n$ and $P(S_{1,n})$ is the maximum probability of segmenting a sentence of n characters into one or two-character words. The probability of a one-character word (i.e., unigram) is estimated by $P(C_i) = \frac{N(C_i)}{N}$, and the probability of a two-character word (i.e., bigram) is estimated by $P(C_iC_j) = \frac{N(C_iC_j)}{N}$, where $N(C_i)$ is the number of times that character C_i occurs in the corpus, $N(C_iC_j)$ is the number of times that string C_iC_j occurs in the corpus and N is the total number of times that any single character terms and any two-character terms occurs in the corpus. A sentence is broken into one or two-character terms using the most likely segmentation. For example, for the sentence of three characters, $S = C_1C_2C_3$, the probability of the sentence with the three different possible ways of segmentation are given, respectively, by

$$\begin{aligned} P(S,(1,1)) &= P(C_1)P(C_2)P(C_3) && (3) \\ P(S,(1,0)) &= P(C_1)P(C_2C_3) && (4) \\ P(S,(0,1)) &= P(C_1C_2)P(C3) && (5) \end{aligned}$$

Assume that the second segmentation method ($k = (1,0)$) has the highest probability, then we break sentence S into C_1/C_2C_3. This is the method we used to break the Chinese sentences in the test collection into one or two-character terms. The probability of a one-character or two-character term is estimated using their occurrence statistics collected from the test documents. When we use this method to segment topics, we assign a small probability to the terms missing in the test collection. The estimated probability for a new term is one over the total number of unique unigrams and bigrams.

3 Test Collection

The TREC-9 CLIR test collection consists of 25 new topics and 127,938 documents from three newspapers, namely the Hong Kong Commercial Daily, Hong Kong Daily News, and Takungpao. The topics are written in English with Chinese translations and contain *title*, *description*, and *narrative* fields.

One of the bilingual dictionaries we used to translate English queries is the Chinese-to-English wordlist (version 2.0) compiled by Linguistic Data Consortium. We downloaded the bilingual wordlist from http://morph.ldc.upenn.edu/Projects/Chinese/. The wordlist consists of a list of Chinese words, paired with a set of English words. The wordlist has some 128,000 entries.

The other bilingual dictionary used in our experiments is the online KingSoft dictionary at http://ciba.kingsoft.net/online/. It consists of a general dictionary and a set of 23 specialized dictionaries, such as ships, electricity, telecommunication, law, broadcasting, environment, chemistry, economy and trade, computer, medicine, and so on. The general dictionary contains about four million entries and the specialized dictionaries together contain about two million entries [6].

4 Monolingual Experiment

The Chinese documents and the Chinese translations of the English topics were indexed using the overlapping bigram technique. All three fields – title, description, and narrative – in the topics were used. The retrieval performance of the monolingual run BRKCCA1 is presented in the second column in table 1. The overall precision is 0.2936 and recall is .855.

5 Cross-Language Retrieval

There are a number of ways to perform the task of cross-language information retrieval in which a query posed in one language is searched against a collection of documents written in a different language. Oard and Diekema provide a recent survey on cross-language information retrieval in [8]. It is obvious that any retrieval method based on matching a query in one language against documents in a different language would fail when there are no cognates between this language pair (e.g., Chinese and English). For matching-based retrieval algorithms to work, both the documents and queries need to be expressed in the same language or conceptual space as in the latent semantic indexing. A common approach to cross-language information retrieval is to couple translation with monolingual information retrieval. One can translate users' queries into the document language, or translate documents into the query language, or translate both the queries and documents into a third language. one can translate queries or documents using a machine translation system. When such resource is not available, one can use bilingual dictionaries, if available, to do word translation or phrase translation, or one can resort to parallel or comparable bilingual corpora from which to mine translation dictionary for cross-language retrieval.

For the English-Chinese cross language retrieval experiments reported below, we take the simple approach of translating queries to the document language, that is, we translate the English queries into Chinese. We then apply the monolingual retrieval ranking algorithm to rank Chinese documents by their estimated probability of relevance to the translated Chinese queries.

5.1 Topics Preprocessing

The topics were processed in three steps to generate the queries before translation. First, the topics were tagged using Brill's part-of-speech tagger [1]. Second, noun phrases are extracted from the tagged topics. Third, the single-word terms and phrases are normalized using a morphological analyzer. The following text shows the tagged text of the description field in topic CH58.

> Are/VBP environmental/JJ protection/NN laws/NNS being/VBG enforced/VBN in/IN China/NNP and/CC Hong/NNP Kong/NNP ?/.

Each word is followed by its part-of-speech tag. The tags NN and NNS represent singular nouns and plural nouns, respectively; NNP represents the proper name, and JJ represents adjective. Then the tagged text is passed to a noun phrase recognizer for noun phrase extraction. The recognizer detects simple noun phrases based on the pattern of the tags. The noun phrase patterns we used to extract noun phrases can be concisely specified in a three-state automaton as shown in Figure 1. The initial state is 0 and the final state is 2. Any words tagged with part-of-speech tags NN, NNS, NNP, NP and NPS are represented by the label NOUN, and words tagged with JJ, JJR, and JJS, which are the positive, comparative and superlative form of an adjective, are represented by the label ADJ. Any sequence of words whose part-of-speech tags completes a path from the initial state to the final state will be extracted as a noun phrase, excluding the single-word nouns.

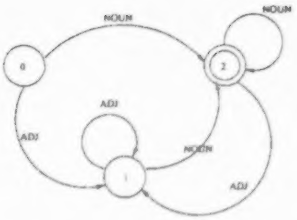


Figure 1. Simple noun phrase automaton

The noun phrases extracted from the above tagged text are *environmental protection laws* and *Hong Kong*. The words appearing in the stoplist were removed and then the remaining single words and noun phrases are normalized using a morphological analyzer [3], which reduces plural nouns to their singular form and verbs to their base form. Also, all words and phrases are converted to lower case. The normalized single words and the simple noun phrases constitute the English queries before translation.

5.2 Query Translation

After the preprocessing of the English topics, each query now is comprised of single words and noun phrases. We translate each query by looking up every single word and noun phrase in a Chinese-English bilingual dictionary.

For BRKECA1 run, a query term (noun phrase or single word) was looked up in the LDC bilingual wordlists. The top two Chinese translation equivalents that occur most frequently in the test document collection were retained as translations for an English term when there are more than two translations for that term. When there is no exact matching for a single-word term, that term is not translated. However when there is no exact matching for a noun phrase, we proceed to match the sub-phrases against the dictionary until there are some matches. If all sub-phrases matching fails, we then look for exact matching for the component words in the phrase. For example, if a three-word phrase $w_1w_2w_3$ is missing in the dictionary, we will search the sub-phrases w_1w_2 and w_3; and if there is no match for w_1w_2, we will search w_1 and w_2w_3 in the dictionary. If none of the sub-phrases is found in the dictionary, we translate this phrase word-by-word by looking up each component word in the dictionary, and take the Chinese translations of all the component words in the phrase as the translation of the phrase.

The Chinese translation equivalents were then segmented into one or two-character words using the segmentation method as described above. The documents in the collection were segmented into one or two-character words as well.

For BRKECA2 and BRKECM1 runs, the noun phrases and their constituent words were looked up in the online KingSoft Chinese/English dictionary. The first Chinese translation for a phrase or word was retained. The Chinese translation equivalents were segmented into words using the longest-matching method. These two runs used the word-based document index for retrieval.

5.3 Manual Query Reformulation

It has been the policy of the Berkeley group to attempt to create manual reformulations of TREC queries since TREC-2. Manual queries usually result in additional relevant documents found which enriches the value of the collection when used for machine learning in the future. Initially this manual reformulation was done without reference to the retrieval, i.e. by searching a comparable collection using the original topic terms. The first of these was the news title database available as part of the University of California's electronic library catalog. Later, as the TREC rules were relaxed to include manual relevant feedback, we have utilized that technique for finding words from top documents of an initial search or by manually marking particular documents as relevant. These techniques were used in our recent CLEF experiments for European languages.

For TREC-9 we created manual versions of the English queries by searching the WWW with topic words and taking pertinent text from the URLs found and inserting it to the manual version of the query. For example topic CH60 has description "Are China and Taiwan developing any types of laser weapons?" Using the words 'China', 'laser weapons' in a GOOGLE search returns the url: http://www.freerepublic.com/forum/a363ee3c93414.htm which has an initial sentence:

> China's People's Liberation Army is building lasers to destroy satellites and already has beam weapons capable of damaging sensors on space-based reconnaissance and intelligence systems, according to a Pentagon report.

which was incorporated into the manual version of that query. While the precision for our manual run BRKECM1 of .8875 was better than one automatic run BRKECA1 (precision 0.3821), it lagged our other run BRKECA2 (precision 0.9500).

One query for which manual augmentation worked well was topic CH67 "Tiananmen Anniversary on Mainland" which a www news archive provided the following additional sentences:

> On June 21, the SCMP reported the detentions on June 19 of 5 dissidents in Hangzhou. The 5 are ZHU LUFU, HAN SHENDAI, WANG RONGQING, MAO QINGXIANG, and LI BAGEN. The last three have been detained several times already over the the past month or two. The 5 are members of the China Democracy Party. Information Centre of Human Rights and Democratic Movement in China says that over 180 CDP members have been arrested in the past month, and 31 are still in detention and awaiting trial.

and

> the Free China Movement describing the arrest and sentencing of ZHOU YONGJUN. Zhou snuck into China in December to visit his parents. Zhou was jailed for two years after the 1989 Tiananmen massacre and subsequent crackdown. After his release 7 years ago he was exiled.

The performance of this manual query increased ten-fold to 0.2009 over the median precision of .0290 and our automatic run precisions of 0.0026 and 0.0378.

Another query, CH79, "Livestock in China". A GOOGLE search "China livestock" yielded a url at Cornell University: http://usda.mannlib.cornell.edu/datasets/international/90014/ which offered statistical information on China's agriculture production. Its descriptive sentences:

> Comprehensive data on Chinese animal agriculture including production of red meats, milk, eggs, poultry meats, and honey by region and province. Also includes inventory data on cattle, hogs, sheep, goats, and draft animals.

were added to the manual query. The performance of BRKECM1 for topic CH79 was 0.1496, almost three times better than our best automatic run BRKECA2 (0.0545).

Overall, the precision of the manual run over 25 topics was 0.1869. This was 28 percent better than the average of medians for topics and 10.2 percent better than our best automatic run (BRKECA1, overall precision 0.1680).

The use of web searches and direct cut-and-paste transfer of new query words and sentences made manual reformulation quite fast. Our estimate is that an average of 10 minutes per query was spent on manual rewrite, or slightly more than four hours total.

5.4 Experimental Results

We performed three English to Chinese cross-language retrieval runs. The title, description, and narrative fields were used in all three runs. For BRKECA1, the queries were translated into English by LDC dictionary lookup. The Chinese translation equivalents were then segmented into non-overlapping bigrams and unigrams. The evaluation result for the BRKECA1 run is presented in the third column in table 1. The evaluation results for BRKECA2 and BRKECM1 are presented in in column 4 and 5 in table 1. The Chinese translation equivalents for these two runs were segmented into words using the longest-matching method. And the segmented Chinese queries were searched against the test document collection which was also segmented into words using the same method. The best automatic English-

recall level	BRKCCA1 (MONO)	BRKECA1 (CLIR)	BRKECA2 (CLIR)	BRKECM1 (CLIR)
at 0.00	0.7079	0.4296	0.3603	0.5624
at 0.10	0.4697	0.3325	0.2828	0.3561
at 0.20	0.4047	0.2655	0.2071	0.2900
at 0.30	0.3720	0.2306	0.1852	0.2264
at 0.40	0.3225	0.1763	0.1555	0.1878
at 0.50	0.2769	0.1586	0.1393	0.1523
at 0.60	0.2445	0.1338	0.1269	0.1261
at 0.70	0.2165	0.1062	0.1052	0.1042
at 0.80	0.1874	0.0664	0.0892	0.0946
at 0.90	0.1368	0.0526	0.0833	0.0851
at 1.00	0.1155	0.0417	0.0721	0.0748
average precision	0.2936	0.1680	0.1543	0.1869
relevant retrieved	567	465	384	451
% of mono		57.22%	52.55%	63.66%

Table 1. Evaluation results for one Chinese monolingual run and three English to Chinese cross-language retrieval runs.

Chinese cross-language retrieval performance is only about 57% of the monolingual retrieval performance. For 5 out of the 25 topics, the precision for the cross-language retrieval is higher than that for the monolingual retrieval. On the other hand, for 10 out of the 25 topics, the precision for the cross-language retrieval is much lower than that for the monolingual retrieval. The main reason is that some key concept terms in those topics were either not translated at all due to the limited coverage of the bilingual wordlist we used or improperly translated. For example, the monolingual precision is .5406 for topic CH78, but the cross-language precision is only 0.0037 for the same topic. Topic CH78 is about motor vehicle fatalities in China. A key concept term 'fatalities' was not translated because it is missing in the LDC dictionary we used. The term 'silk' in topic 74 was translated into 绸, instead of the more appropriate term " 丝绸 ". For topic CH63, the noun phrase 'energy source (能源)' was translated into two Chinese words, 能 (energy) and 源 (source). The main concept term 'three-links (三通)' in topic CH70 were translated word-by-word into 三 (three) and 连接 / 相连 (link). Not being able to translate the term 'industrially' and mistranslating the term 'developed' in topic CH72 resulted in very lower precision in cross-language retrieval. The precision per topic for the monolingual run and the three English-Chinese cross-language runs are presented in table 2.

6 Conclusions

In summary, we performed three English-Chinese cross-language information retrieval runs, one manual and two automatic. We took a simple approach of translating queries into document language by dictionary lookup in our cross-language retrieval experiments. Even though the dictionary used in the BRKECA2 run is much larger than the one used in the BRKECA1 run, the retrieval performance for BRKECA2 is slightly worse than that for BRKECA1. We believe the inferior performance can be attributed to the simple selection method and to the difference in word usages. The performance of the best automatic run is only about 57% of the monolingual performance. The main performance-limiting factor is the limited coverage of the dictionary used in query translation. Some of the key concepts were either not translated or improperly translated.

Topic No	BRKCCA1 (MONO)	BRKECA1 (CLIR)	BRKECA2 (CLIR)	BRKECM1 (CLIR)
CH55	0.2200	0.1757	0.0973	0.1382
CH56	0.2814	0.1270	0.2293	0.1928
CH57	0.2939	0.1348	0.1435	0.1386
CH58	0.0036	0.0022	0.0089	0.0059
CH59	0.0015	0.0000	0.0000	0.0000
CH60	1.0000	0.3821	0.9500	0.8875
CH61	0.0000	0.0124	0.0445	0.0115
CH62	0.5000	0.0909	0.0032	0.0019
CH63	0.3009	0.0001	0.0114	0.1118
CH64	0.5354	0.3196	0.3128	0.3840
CH65	0.1797	0.7058	0.0453	0.0133
CH66	1.0000	0.8333	1.0000	1.0000
CH67	0.1327	0.0378	0.0026	0.2009
CH68	0.1865	0.0165	0.0066	0.0738
CH69	0.1497	0.2916	0.0329	0.0531
CH70	0.1687	0.0057	0.0001	0.0025
CH71	0.2604	0.1456	0.0467	0.2768
CH72	0.2910	0.0314	0.0755	0.1174
CH73	0.1966	0.3311	0.0004	0.0789
CH74	0.2655	0.0004	0.5286	0.3772
CH75	0.1413	0.2922	0.1102	0.2883
CH76	0.5065	0.2140	0.1460	0.1495
CH77	0.0434	0.0263	0.0029	0.0043
CH78	0.5406	0.0037	0.0033	0.0145
CH79	0.1417	0.0188	0.0565	0.1496

Table 2. Precision per topic for the monolingual run and three English-Chinese cross-language runs.

7 Acknowledgements

This research was supported by DARPA (Department of Defense Advanced Research Projects Agency) under research grant N66001-00-1-8911 as part of the DARPA Translingual Information Detection, Extraction, and Summarization Program (TIDES).

References

[1] Eric Brill. A simple rule-based part of speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, 1992.

[2] W. S. Cooper, A. Chen, and F. C. Gey. Full text retrieval based on probabilistic equations with coefficie nts fitted by logistic regression. In D. K. Harman, editor, *The Second Text REtrieval Conference (TREC-2)*, pages 57–66, March 1994.

[3] M. Zaidel D. Karp, Y. Schabes and D. Egedi. A freely available wide coverage morphological analyzer for english. In *Proceedings of COLING*, 1992.

[4] Y. Dai and T. Loh. A New Statistical Formula for Chinese Text Segmentation Incorporating Contextual Information. In *SIGIR'99, Berkeley, August 1999*, pages 82–89, 1999.

[5] X. Ge, W. Pratt, and P. Smyth. Discovering Chinese Words from Unsegmented Text. In *SIGIR'99, Berkeley, August 1999*, pages 271–272, 1999.

[6] KingSoft. http://ciba.kingsoft.net/ciba2000/cidian.htm.

[7] J. Nie and F. Ren. Chinese information retrieval: using characters or words? *Information Processing and Management*, 35:443–462, 1999.

[8] D. Oard and A. Diekema. *Cross-Language Information Retrieval*, volume 33, pages 223–256. 1998.

[9] M. Sun, D. Shen, and C. Huang. CSeg&Tag1.0: A Practical Word Segmenter and POS Tagger for Chines e Texts. In *Proceedings of the Fifth Applied Natural Language Processing Co nference*, pages 119–126, 1997.

Question answering, relevance feedback and summarisation: TREC-9 interactive track report

Neill Alexander, Craig Brown, Joemon Jose, Ian Ruthven[1] and Anastasios Tombros
Department of Computing Science
University of Glasgow, Glasgow, G12 8QQ. Scotland
alexannw,brownca,jj,igr,tombrosa@dcs.gla.ac.uk

Abstract

In this paper we report on the effectiveness of query-biased summaries for a question-answering task. Our summarisation system presents searchers with short summaries of documents, composed of a series of highly matching sentences extracted from the documents. These summaries are also used as evidence for a query expansion algorithm to test the use of summaries as evidence for interactive and automatic query expansion.

1. Introduction

The main focus of Glasgow's Interactive Track study was to investigate the use of summaries in interactive searching.

Our experiments used a form of the query-biasing summarisation technique proposed by Tombros and Sanderson [TS98], to create short document summaries that are tailored to the user's query. These summaries highlight the main points of the document that pertain to the query. The summaries are based on highly matching sentences, allowing users to view the context in which query terms are used within the document.

We hypothesised that this form of summarisation would be particularly effective for the time-limited, query-answering task of the interactive track, in that summaries would allow users to filter out non-relevant documents more effectively and target potentially relevant documents more quickly than either title alone or the full text of the documents.

In addition, we investigated the use of summaries for relevance feedback (RF): by using the content of the summaries, rather than the full-text of the documents, to generate query expansion terms.

Our experiments indicate that although RF is generally not considered helpful, summaries can provide a popular and useful aid to finding relevant information.

The paper is structured as follows: in section 2 we describe the system we used in these experiments including details of the summarisation process, in section 3 we describe the interface, in section 4 we give details of the experimental subjects and in section 5 we analyse the results. We conclude in section 6.

---

[1] Corresponding author.

2. System

In section 2.1 we outline the main components of our system and in section 2.2 we describe how the summaries are created.

2.1 System architecture

Our experimental system was composed of three units:

1. *retrieval system*. The retrieval system (SMART) performs an initial query run using the query terms passed from the interface. The list of retrieved document identifiers and document titles are passed to the interface for display.
2. *summariser*. For each retrieved document, the summariser (described in section 2.2) generates a query-biased summary, which is passed to the interface on demand.
3. *interface*. The interface displays the retrieved document identifiers, document titles and summary. The overall look and feel of the interface is described in section 3.1. The interface is also responsible for logging user interaction and generating query expansion terms (described in section 3.2).

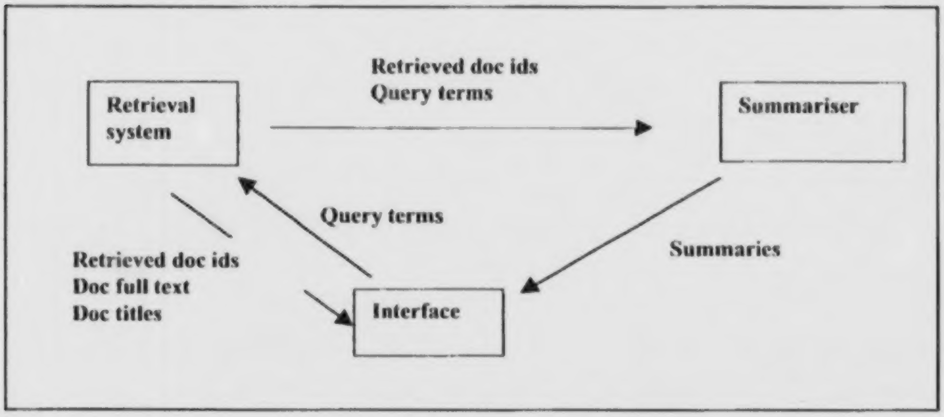


Figure 1: System architecture

2.2 Summariser

A document summary conventionally refers to a condensed version of a document that succinctly presents the main points of the original document. *Query-biased* summarisation methods generate summaries in the context of an information need expressed as a query by a user. Such methods aim to identify and present to the user individual parts of the text that are more focused towards this particular information need than a generic, non-query-sensitive summary. In this way summaries can serve an *indicative* function, providing a preview format to support relevance assessments on the full text of documents [RSZ71].

Query-biased text summarisation is an emerging area of summarisation research that had not been addressed until recently. Tombros and Sanderson looked into the application of such methods in information retrieval, evaluating the indicative function of the summaries [TS98]. Their study showed that users were better able to identify relevant documents when using the summaries than when using the first few sentences of a document. Recently the TIPSTER funded SUMMAC project [MHKHOFCS98] provided a framework for the evaluation of different types of summarisation systems. As part of that project, a number of query-biased summarisation systems were evaluated by measuring their ability to help users identify documents relevant to a query.

The summaries generated by our system were indicative and query-biased, aiming to provide users working on an interactive IR system with information on the relevance of documents retrieved in response to their query. The system is based on a number of sentence extraction methods that utilise information both from the documents of the collection and from the queries submitted, and is a simplified version of the system described in [TS98].

Each document that was retrieved in response to a specific query was passed through the summarisation system, and as a result a score for each sentence of each document was computed. This score represents the sentence's importance for inclusion in the document's summary. Scores are assigned to sentences by examining the structural organisation of each document, and by utilising the inverse document frequency (IDF) weights assigned to each term. Information from the structural organisation of the documents was utilised in two ways. Terms occurring in the title section of a document were assigned a positive weight (title score) in order to reflect the fact that headlines of news articles tend to reveal the major subject of the article. In addition, a positive ordinal weight was assigned to the first two sentences of each article, capturing the informativeness of the leading text of news articles. The IDF weights of the terms in the collection were used as a source of evidence of attributing an overall measure of importance for each sentence of the source documents. In order to establish such a measure, the sum of the IDF weights of all the terms comprising a sentence was divided by the total number of terms in that sentence. In that way an importance score was attributed to each sentence in the collection.

In addition to the scores assigned to sentences, information from the query submitted by the user was also employed in order to compute the final score for each sentence. A query score was thus computed, intended to represent the distribution of query words in a sentence. The rationale for this choice was that, by allowing users to see the context in which the query terms occurred, they could better judge the relevance of a document to the query. The computation of that score was based on the distribution of query terms in each sentence. This was based on the belief that the larger the number of query terms in a sentence, the more likely that sentence conveyed a significant amount of the information need expressed in the query. The actual measure of significance of a sentence in relation to a specific query, was derived by dividing the square of the number of query terms included in that sentence by the total number of the terms comprising the query.

The final score for each sentence is calculated by summing the partial scores discussed above. The summary for each document is then generated by selecting the top-scoring sentences, and outputting them in the order in which they appear in the original document. Summary length was defined to be 20% of the document's length, up to a maximum of 6 sentences. Such a value seems to be in general agreement with suggestions made by [Ed64, BMR95].

Figure 3 shows the summary produced from the document in Figure 2, retrieved in response to the query '*America national parks redwood trees*'. In Figure 2 bold type marks those sentences that were extracted to form the summary.

<DOC>
<DOCNO> SJMN91-06312178 </DOCNO>
<ACCESS> 06312178 </ACCESS>
<DESCRIPT> CALIFORNIA; TREE; PARK; US </DESCRIPT>
<LEADPARA> **California's majestic redwood parks may be ceded to the federal government under a cost-cutting proposal under study by state Parks and Recreation Department officials, the officials said Wednesday**.; The three parks -- Jedediah Smith Redwoods State Park and Del Norte Coast Redwoods State Park in Del Norte County, and Prairie Creek Redwoods State Park in Humboldt County -- are the crown jewels of the state park system and home to 2,000-year-old redwoods, among the oldest living things on Earth. </LEADPARA>
<SECTION> California News </SECTION>
<HEADLINE> STATE MAY CEDE ITS 3 REDWOOD PARKS TO U.S. </HEADLINE>
<TEXT> The proposal emerged from a review ordered by state Parks and Recreation Director Henry R. Agonia after the Wilson administration sent a directive to, state agencies asking them to identify budget cuts. The state is facing a staggering $2 billion deficit in this year's $55.7 billion budget.; The prospect of transferring California's redwood parks to the National Park Service drew praise and criticism from environmentalists and park rangers Wednesday as word spread.; "I'm strongly against it," state parks Superintendent Bill Beap said in a telephone interview from Eureka. "These are the prime jewels of the state park system."; But the proposal was welcomed by Sierra Club officials, who called it "a splendid idea." **Edgar Wayburn, the club's vice**

president for conservation, noted that the neighboring National Redwood Park's boundaries touch all three state parks. </TEXT>
<BYLINE> Los Angeles Times </BYLINE>

Figure 2: Sample document

<SUMMARY>
<TITLE> STATE MAY CEDE ITS 3 REDWOOD PARKS TO U.S.
</TITLE>
<DOCID> SJMN91-06312178 </DOCID>
<LI>California's majestic redwood parks may be ceded to the federal government under a cost-cutting proposal under study by state Parks and Recreation Department officials, the officials said Wednesday. </LI>
<LI> " Edgar Wayburn, the club's vice president for conservation, noted that the neighboring National Redwood Park's boundaries touch all three state parks. </LI>
</SUMMARY>

Figure 3: Summary produced from document SJMN91-06312178

3. Interface

3.1 Look and feel

The interface consists of four main components: list of 20 retrieved document titles and associated check boxes for marking documents relevant, summary display, query box and suggested expansion terms.

A summary was generated on demand each time the user moved the mouse pointer over a document title. The complete document could be viewed, in a separate window, by clicking on the document title.

Query expansion was both automatic (the top 6 expansion terms were automatically added to the query when the user requested more documents), and interactive. The user could request suggestions for new query terms by clicking the 'Get More Terms' button, the suggested terms appearing in the 'More Terms' box on the bottom left of the screen. Each query, and suggested, term was displayed in this box; check-boxes were used to add/delete terms from the query.

3.2 Relevance feedback

To assist users in modifying their queries, a query expansion facility was offered. Expansion terms were selected from the summaries of marked relevant documents using Porter's term weighting function [PG88]. This is shown in Equation 1, where r equals the number of relevant summaries containing a term, R is the number of relevant documents, n is the number of documents in the collection containing the term, and N is the number of documents in the collection.

$$Porter = {}^{r}\!/_{R} - {}^{n}\!/_{N}$$

Equation 1: Porter term weighting function

This use of Porter's term weighting function differs from other applications in that r is based on the terms appearance in a summary of a relevant document, rather than the whole document. The basis for this is that a query-biased summary of a document may be a better source of relevant terms than the full-text of the document. This may be especially true for long documents that cover many topics.

For this experiment we used only the summary to determine possible expansion terms, even if the user had viewed the full-text of the document. A natural enhancement of this may be to vary the source of the expansion terms according to the representation(s) of the document viewed by the user. For example if the user has only viewed a summary before making an assessment, then the summary should be used for feedback, if the user has viewed the full text of the document, then the full-text may be a better source of evidence for feedback.

This method of calculating relevance weights has previously been shown to give good results [PG88, Efth95] for automatic and interactive query expansion. The expansion terms produced in our case, however, were not always useful. This was for two reasons: low numbers of relevant documents, and the processing of summaries for relevance weighting.

The searchers tended to find relatively low numbers of relevant documents. If a searcher only found one relevant document in the first display of documents (as many did) then the relevance weighting prioritised those terms that only appeared in the relevant document. In this case, not only were these terms not useful for retrieving further relevant documents but occasionally the terms turned out to be spelling errors in the original documents, e.g. '*armioffici*', '*withth*', and '*sovietunion*'.

A flaw in our preparation of the summaries for relevance weighting was not to remove stop words from the summaries before weighting. Consequently, a proportion of the suggested expansion terms consisted of labels such as, '*docno*' and '*bylin*'. This was shown to affect a small number of queries.

4.Experimental details

4.1 Subjects

All subjects were educated to graduate level in a non-computing, non-LIS discipline, and, with two exceptions, all our subjects were postgraduate students recruited from the Information Technology course at Glasgow University. None of the subjects had any formal training in information searching or retrieval, beyond basic training on the university library search facilities.

The average age of the subjects was 23 years, and the average previous search experience was 3 years. All subjects reported some experience with library systems but the majority of reported experience was gained using web search engines using a point-and-click interface. None of the subjects had used either the control system (ZPRISE), the experimental system or an IR system with summarisation facilities.

These subjects were relatively regular searchers, performing searches either daily or weekly, but were neutral as to how much they enjoyed the process of searching for information.

4.2 Technical drawbacks

There were two main technical problems observed in our experiments, both related to the length of time taken to produce summaries.

Summaries are produced on request, section 3.1, consequently speed is an important issue for this system. The average time taken to produce a summary was 5 seconds, the range being 0.5-20 seconds. However in practice summaries took longer than average to generate, somewhere of the order of 10 seconds on average. The main reason for this is that the summarisation time is dependent on the length of the original document.

In our experiments, the retrieved documents tended to be longer than average, consequently the summaries took longer than average to produce. This was criticised by several users, as discussed in section 5.2.

Secondly, summaries were requested by running the mouse pointer over a document title. This method was intended to be an intuitive method of obtaining summaries. However, as users were unfamiliar with the interface, they often scanned the pointer over several document titles unintentionally. This would not be a problem if summaries were created instantaneously but, as summaries took longer than expected to create, this often had the unfortunate side-effect of halting the interface until the summaries for the scanned documents had been produced. The user had then to wait until all summaries were created before being able to issue any more commands.

A further, although minor, problem was that as soon as the user moved the pointer away from a document title, the summary disappeared. This was criticised by several users who would have preferred the summary to remain visible until a new summary was requested.

5. Analysis

The experiments were run according to the matrix supplied by the track organisers, however technical problems meant that we were unable to run the complete set of subjects. Our final submission consisted of the full results for 10 out of 16 subjects[2]. Hence the following analysis is partial.

5.1 Quantitative analysis

5.1.1 Interface

To examine the novel features of our interface we extended the TREC post-system questionnaire to include specific questions on the use of summaries and RF.

The subjects were, on average, in favour of the summaries with an average score of 3.7[3] for the question "*Were the document summaries useful in answering the questions?*" and 3.5 for the question "*Were the document summaries a good representation of the full document text?*". However, it is doubtful whether this second result was valid, as few users actually compared the full-text with the summary.

9 of 10 subjects judged the length of the summary as "*About right*", the remaining subject said the summaries were too short.

The subjects were less convinced about the benefits of RF with an average response of 2.7 for the question "*Was relevance feedback useful?*", 2.1 for the question "*Did the system add good terms to your query?*" and 2.5 for "*How well did you understand the relevance feedback option?*".

RF was also shown to be unpopular in the exit questionnaires, we shall discuss this in more detail in section 5.2.

5.1.2 Topics

The subjects were generally unfamiliar with the topics before searching, the reported certainty before searching being on average between 1.24 – 1.9. The certainty of the users increased after searching for all topics. The final certainty ranged from 2.97 for topic 6, to 4.4 for topic 1. Although the pre-search certainty for both types of topics were roughly the same (1.60 for multiple part topics, 1.59 for comparison topics), searchers reported a greater degree of post-search certainty for multiple part topics (3.51 multiple part against 3.15 for comparison topics).

Although users found slightly more relevant documents for the multiple part topics (1.04 per search vs. 1.01 for comparison topics), the greater reported certainty seems to come from the interaction rather than search success. For the multiple part topics, subjects reported that these topics were easier to start a search on, and easier to search for[4]. However they reported that they were less satisfied with searches on the multiple part topics (3.00 vs. 3.27 comparison) and would have liked more time to search on these topics (3.41 multiple part vs. 3.45 comparison). There was a greater pre-search familiarity with the multiple part topics (1.96 vs. 1.78) not reflected in pre-search certainty.

5.1.3 Systems

Users marked slightly more relevant documents on average with the control system (1.04 vs. 1.01 per query), found it easier to start a search with the control system (3.58 vs. 3.36) and easier to search (3.05 vs. 3) but were overall less satisfied with the control system (2.62 vs. 2.51). The subjects felt they would have preferred more time with experimental system (2.62 vs. 2.81), possibly due to the time delay in producing the summaries.

From the exit questionnaires, the subjects claimed a relatively high level of understanding of the task (4.4), a fair similarity with other searching tasks (3.5). The subjects did not feel there was a great difference between systems (3.4).

---

[2] Five subjects started searching on the control system, five on the experimental system.

[3] These averages, and others in the remainder of the paper, are out of a possible 5, taken from the answers given in the standard TREC questionnaires.

[4] Ease to start (multiple part 4.46 vs. 3.97 comparison), ease to search (3.77 multiple part vs. 3.54 comparison).

Of the ten subjects tested 6 claimed the experimental system was easier to learn to use (1 for ZPRISE, 3 undecided), 7 found the experimental system easier to use (3 ZPRISE, none undecided), and 7 preferred the simplicity of the experimental system (3 opting for ZPRISE).

5.1.4 Search statistics

In this section we analyse the search statistics regarding the number of documents retrieved, query terms entered and documents assessed relevant for both systems. Table 1 summarises the basic search statistics.

	Control System	**Experimental System**
Initial query terms	3.54	4.69
Query terms added	1.24	1.55
Iterations (including initial ranking)	1.5	2.16
Documents assessed relevant by subject	1.04	1.01
Full texts viewed	2.94	1.19
Summaries generated	-	5.48

Table 1: Average search statistics per query

The searchers tended to use more query terms on the experimental interface than the control system and more terms were added through query expansion. Very few terms were added through the interactive query expansion facility. As noted before the searchers assessed slightly more relevant documents with the control system than the experimental system.

The searchers appeared to do more iterations of feedback with the experimental system although this is slightly deceptive as in fact they tended to do a new search more often than modify their existing query.

5.1.5 Search results

In Table 2 we outline the overall search results for our experiment. The users returned a higher proportion of non-supporting documents on the control system (41.67% - sum of columns 4, 7 and 8 in Table 2) than on the experimental system (36.84%), and a lower proportion of supporting documents with the control system (58.33% control vs 63.16% experimental – sum of columns 2 and 5 in Table 2).

	2:2	**2:1**	**2:0**	**1:2**	**1:1**	**1:0**	**0:0**
Control responses	8	-	-	6	-	1	9
Control average	33.33%	-	-	25.00%	-	4.17%	37.50%
Experimental responses	6	-	1	6	-	2	4
Experimental average	31.58%	-	5.26%	31.58%	-	10.53%	21.05%

Table 2: Summarised results – control system versus experimental system

		2:2	**2:1**	**2:0**	**1:2**	**1:1**	**1:0**	**0:0**
Topics 1 -4	**TREC average**	13.91%	2.16%	0.96%	28.06%	9.35%	9.83%	35.73%
	Our average	4.00%	-	-	48.00%	-	12.00%	36.00%
Topics 5 - 8	**TREC average**	46.60%	-	14.56%	-	-	-	38.84%
	Our average	72.22%	-	5.56%	-	-	-	22.22%

Table 3: Summarised results – Glasgow results versus average results

In Table 3 we compare our results from both systems with the average from the TREC participants. For the multiple part topics (1-4), our subjects returned a smaller percentage of fully supporting documents than average but a higher percentage of documents that supported a partial answer. The subjects also returned slightly more than average non-supporting documents.

For the comparison topics (topics 5 – 8) the subjects returned noticeably higher percentage of supporting documents and fewer non-supporting documents, suggesting that these topics were easier for our subjects, although in both our control and experimental systems our searchers returned more documents for the multiple part topics.

Topic	Control unique	Control total	Experimental unique	Experimental total	Total unique for query	Total TREC	Averages
1	1	5	1	3	1	13	7.62%
2	1	1	-	-	1	7	14.29%
3	2	6	2	5	3	17	17.65%
4	2	3	9	11	11	39	28.21%
5	3	7	3	4	5	7	71.43%
6	2	2	2	5	2	3	66.67%
7	5	9	1	1	6	23	26.09%
8	1	1	-	-	1	15	6.67%
Total	17	34	17	29	30		
Average	50.00%		58.62%				

Table 4: Document analysis results

In Table 4, we present an analysis of the overlap of the supporting documents found with the two systems. These results attempt to capture the performance of subjects using either system to discover new documents that support the answer to each query. Any document returned by a subject for a specific query that was marked by TREC assessors as 'supporting the right answer for this query' was used in the calculations, irrespective of the score assigned to that response. That means that even if a subject provided a wrong answer for a query based on a document marked as 'supporting' by the assessors, that subject would still be credited with finding a document that supports the correct answer for that query, and would be included in the calculations.

The **Control unique** and **Experimental unique** columns indicate the number of unique supporting documents found by subjects for each of the two systems (Control or Experimental) for a specific query. The **Control total** and **Experimental total** columns indicate the total number of documents marked by subjects for a specific query for each of the two systems. The **Total unique for query** column displays the total number of unique supporting documents for each query returned by both systems. Finally, the **Total TREC** column indicates the number of documents for each query that were marked as 'supporting the right answer' from the TREC assessors.

The results indicate that, on average, subjects using the experimental system performed better at discovering documents that could potentially support the right answer for a query. Subjects under both systems discovered the same number of unique supporting documents (17), however subjects using the experimental system discovered these documents in fewer attempts (29 vs. 34).

5.1.6 Search narrative

A question searchers have to answer is: *Which Children's TV program was on the air longer: the original Mickey Mouse club or the original Howdy Doody Show?*

The searcher starts with the query containing terms, *children's, TV, programs, Mickey, Mouse, Club, Howdy, Doody, Show.* That is basically used all the terms in the question description itself, suggesting the subject relied heavily on the task given. The system returned twenty documents titles and the searcher went through the list of titles in order.

The searcher viewed summaries of eighteen documents, these were not selected in the same order as the retrieved list. In addition, the searcher viewed the full text of three documents out of the twenty displayed.

There was no relevance feedback iteration and no query modification. The users returned the answer to the query as the Mickey Mouse show ran longer, based on the document, LA050690-0059. The confidence of the user in the answer is Neutral.

5.2 Qualitative

Relevance feedback was not popular amongst our searchers. As noted in the previous section searchers on the experimental system tended to enter new search terms rather than work with the modified query. We believe that the high familiarity with web search engines in our subjects may have promoted this behaviour. The subjects also claimed a poor understanding of relevance feedback. Each subject used the interactive option at least once. The RF option was also relatively unpopular on the control system.

Topic 1, "*What are the names of three US national parks where one can find redwoods*?" caused our subjects some difficulties as eight of the ten subjects returned document SJMN91-06312178 (shown in Figure 2), which discusses State rather than National Parks. The text of this document does make the distinction between the two categories of park, but we doubt whether the subjects read this document closely enough to pick up on this.

In answering the comparison topics, such as "*Which painting did Edvard Munch complete first: "Vampire" or "Puberty"?* ", several subjects did not find any individual document that supplied an answer but managed to find an answer by analysing information found in more than one document. For example, on the Munch topic, several users found a date for the completion of "*Vampire*" in one document, a date for "*Puberty*" in a different document and answered the question citing both documents as evidence.

Although our analysis is not conclusive, we believe that summaries may have resulted in some false positive answers. In this case, the searcher assesses the summary relevant without reading the full text of the (non-supporting) document.

A good example of this is the assessment of document AP890215-0071 for the topic "*Name four films in which Orson Welles appeared*". One summary produced for this document contained the sentence "*Turner Entertainment Co. said it will not colorize Orson Welles' black-and-white film classic "Citizen Kane" because the late director's estate may have the right to prohibit it.*" which correctly identified the film Citizen Kane as being one of the films of Orson Welles.

However the summary also contained the last sentence of the document, "*Movie purists have previously lamented the colorizing of such classics as "It's a Wonderful Life", "Casablanca" and "A Christmas Carol."*" which led the searcher to credit Mr Welles as appearing in these films, even though this was not supported by the text of the full document. Although this may have affected some of our searchers, it may not be a real concern in an operational environment in which users are not so restricted by time limitations.

All subjects liked the use of summaries but felt the summaries took too long to produce. The subjects also liked the simplicity of the interface event if they ignored the RF and query expansion options.

6. Conclusion

Our research aim was to investigate the use of summarisation techniques for a question-answering task. Although our subjects returned a low number of documents, the analysis showed that subjects returned a higher proportion of supporting documents and a lower proportion of non-supporting documents with our summarisation system. The subjects also viewed fewer full documents per relevant document found with the experimental system than the control system. Although our experiments were only partially completed, these two findings indicate that our experimental hypothesis that summaries can help users target relevant documents and eliminate non-relevant documents more effectively is worth investigating further. In addition, the positive response from our subjects towards the use of summaries indicate the summaries are not only effective but can also be a popular aid to searching.

Acknowledgements

Ian Ruthven is currently supported by the Library and Information Commission funded project '*Retrieval through explanation*', Anastasios Tombros is supported by a University of Glasgow Postgraduate Scholarship. We would also like to acknowledge the Computing Science Research Committee for funding this research and all the experimental subjects who took part in these experiments.

References

[BMR95] R. Brandow, K. Mitze, L. and Rau. Automatic condensation of electronic publications by sentence selection. *Information Processing and Management.* **31**. 5. pp 675 - 685. 1995.

[Ed64] H. Edmundson. *Problems in automatic abstracting*. Communications of the ACM. 7. **4**. pp 259 - 263. 1964.

[Efth95] E. N. Efthimiadis. *User-choices: a new yardstick for the evaluation of ranking algorithms for interactive query expansion.* Information Processing and Management. 31. 4. pp 605 - 620. 1995.

[MHKHOFCS98] I. Mani, D. House, G. Klein, L.Hirschman, L. Obrst, T. Firmin, M. Chrzanowski, and B. Sundheim. *The TIPSTER SUMMAC text summarization evaluation: final report.* MITRE Corporation Technical Report. 1998.

[PG88] M. Porter and V. Galpin. *Relevance feedback in a public access catalogue for a research library: Muscat at the Scott Polar Research Institute*. Program. **22**. 1. pp 1 - 20. 1988.

[RSZ71] J. Rush, J, R. Salvador and A. Zamora *Automatic abstracting and indexing. II. Production of indicative abstracts by application of contextual inference and syntactic coherence criteria*. Journal of the American Society for Information Science. 22. **4**. pp 260 - 274. 1971.

[Tom97] A. Tombros. *Reflecting User Information Needs Through Query Biased Summaries*. MSc Thesis, Technical Report (TR-1997-35) of the Department of Computing Science at the University of Glasgow, Glasgow G12 8QQ, UK. 1997.

[TS98] A. Tombros and M. Sanderson. *The advantages of query-biased summaries in Information Retrieval.* Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 2 - 10. 1998.

Filters and Answers: The University of Iowa TREC-9 Results

Elena Catona[1], David Eichmann[2,3] and Padmini Srinivasan[1,2]

[1]Department of Management Sciences
[2]School of Library and Information Science
[3]Computer Science Department
The University of Iowa
{elena-catona, david-eichmann, padmini-srinivasan}@uiowa.edu

The University of Iowa participated in the adaptive filtering and question answering tracks of TREC9. The filtering system used was an extension of the one used in TREC-7 [1] and TREC-8 [2]. Question answering was done using a rule-based system that employed a combination of public domain technologies and the SMART retrieval system.

1 – Adaptive Filtering:

Our approach to filtering involves a two-level dynamic clustering technique. Each filtering topic is used to create a primary cluster that forms a general profile for the topic. Documents that are attracted into a primary cluster participate in a topic-specific second level clustering process yielding what we refer to as secondary clusters. These secondary clusters, depending upon their status, are responsible for declaring, i.e., retrieving, documents for the topic.

As documents are temporally processed they are attracted to a primary cluster if their similarity with the cluster vector is above a primary threshold. These documents enter the secondary clustering stage where again, based on similarity to cluster vectors and a secondary threshold, they either join an existing secondary cluster or start a new one. If at some point the similarity between a secondary cluster and the primary cluster exceeds a third declaration threshold then the document most recently added to the secondary cluster is retrieved for the user.

When deriving representations we use TF*IDF weights after stemming the terms using Porter's stemmer. We also limit document vectors and cluster vectors to the best 100 and 200 stems respectively.

In TREC-8 adaptation was explored at several different levels [2]. First a secondary cluster's future behavior would depend upon past performance. If a secondary cluster declares a document that turns out to be relevant then it is colored green. This means that it declares all documents that join it in the future. If instead the declared document is non relevant then the cluster is colored red and all future documents are not declared. A non relevant document that joins a green cluster spawns an independent red cluster allowing the original cluster to remain green. Another adaptive dimension was to have the primary cluster vector adapt as relevant judgements were obtained. A version of Rochio's feedback method is built into the system for this purpose. A differential adaptation scheme is also built in for this purpose. The key distinction is that in the differential scheme positive and negative term vectors are comprised only of terms not found in the other vector or in the original query vector.

Recent experiments conducted with TREC-8 data explored additional dimensions of adaptation. For example, we experimented with adapting the primary threshold as the performance measure varied. For this, the performance measure such as the utility score was computed at regular intervals when a "snapshot" of the system is taken. We also explored adaptation of secondary and declaration thresholds. In all these the most profitable approach appears to be adaptation of the break threshold using a step function that responds to changes in performance across snapshots. Our OHSU runs use the system as described above with adaptation of the break threshold.

Other key extensions to the system for TREC-9 include the ability to specify the type of index vectors to utilize. A phrase recognizer loads dictionaries of phrases derived from sources such as the WordNet thesaurus and matched phrases are included into the document vectors. More recently a rule-based entity recognizer has been developed that allows the indexing of documents by person names, organizations, locations and events. Our MESH run includes this technology as well special support for medical terminology. The MeSH hierarchy, an associated lexicon of synonyms and a supplementary list of concepts such as drug names were used. The MESH run involved index vectors that were populated using only the entities extracted from the source text.

OHSU Runs:

For TREC-9 we submitted two OHSU runs. These runs employed word based indexing, Rochio feedback for the profile adaptation and adaptation of the declaration threshold. Both OHSU runs used the controlled vocabulary field (MeSH terms). The two runs differ only in their starting threshold values. The primary, secondary and declaration thresholds were 0.3, 0.32, and 0.3 respectively for OHSU1 and 0.25, 0.27, and 0.25 respectively for OHSU2. The declaration threshold was adapted in each case using a step-wise strategy. Figure 1 shows the performance in terms of utility for our OHSU1 run. The dashed bars represent median performance across systems for each topic. There are 24 topics for which OHSU1 was better than the median and another 24 for which it was below the median.

We conducted several experiments after the official submission deadline to better understand the different aspects of our filtering system and its weak performance on the OHSU task. The first question asked was whether the primary filter was effective. In other words how good was it at filtering out non relevant documents while allowing through the relevant documents? Figure 2 shows the percentages filtered through over time, with snapshots taken every 1000 documents. The figure shows that if we divide the snapshots into three groups then the primary filter allows about 50%, 60% and then 59% of the relevant documents that arrive over the first, second and third sequence of snapshots respectively. At the same time the percentage of non relevant documents allowed through stays less than 1% of the number seen. We then examined the effectiveness of the secondary filter. Note that this analysis of the secondary filter was limited to those documents allowed through by the primary filter. Figure 3 shows that the secondary filter was successful in reducing the percentage of non relevant documents allowed through (dashed bars). However, at the same time it also restricts the passage of relevant documents although not as severely. Next we took a different track in our analysis and examined the effectiveness in adapting the declaration threshold. Figure 4 displays these results. We can observe that if we eliminate break threshold adaptation performance degrades significantly over time (dashed bars). In contrast, the adaptive mode is able to

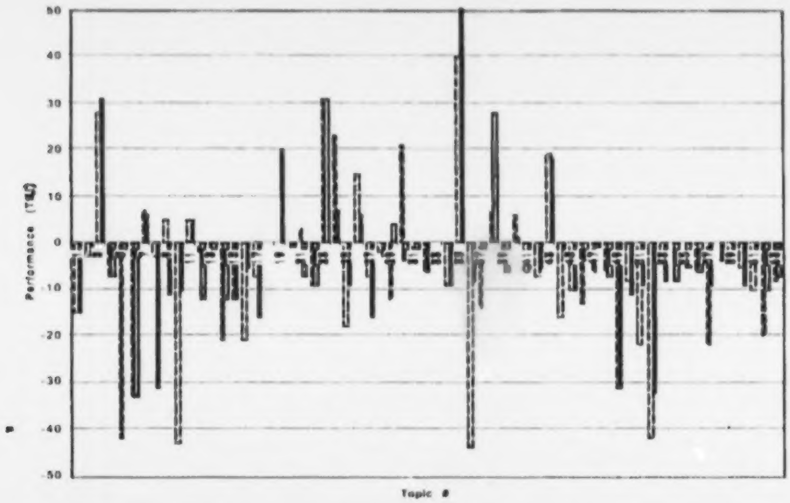


Figure 1: Performance of OHSU1. Dashed bar: OHSU1, Solid bar: median performance

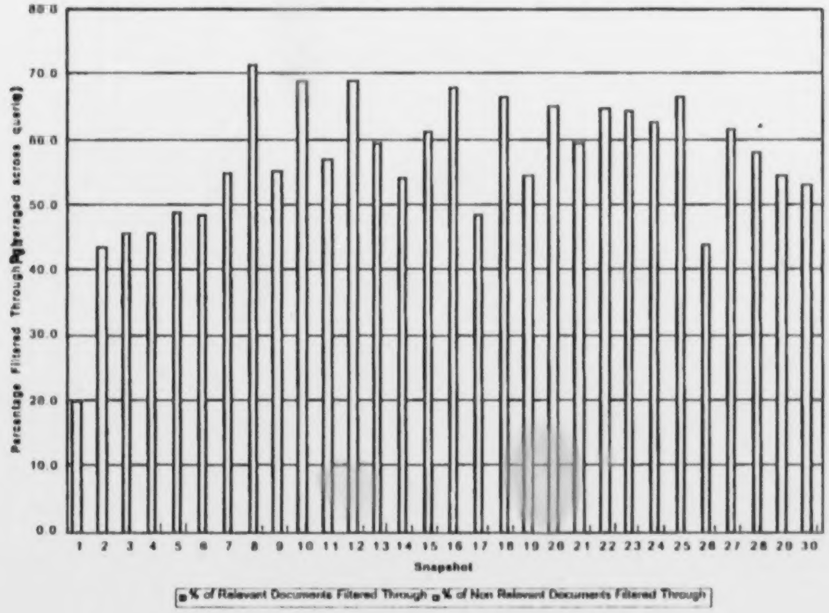


Figure 2: Assessment of Primary Filter

stay somewhat steady - although on the negative side of the performance axis. At this point we suspected that our break threshold may not be restrictive enough. Figure 5 shows the effect of testing this by contrasting a run where the break threshold was increased from the original 0.25 (dashed bars) to 0.3 (solid bars).

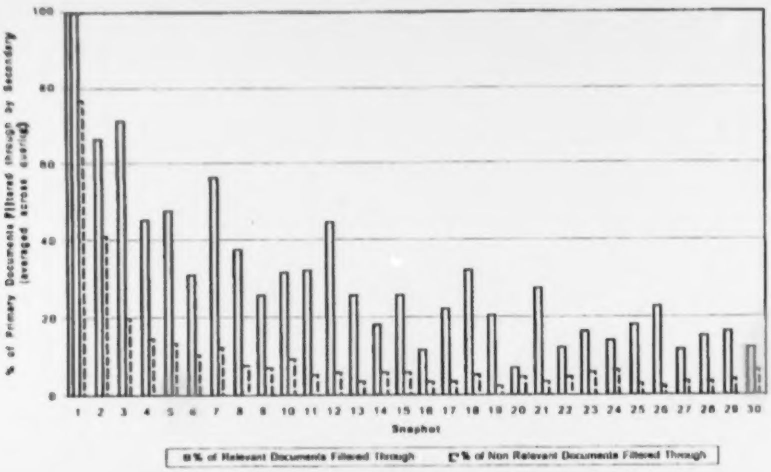


Figure 3: Assessment of Secondary Filter

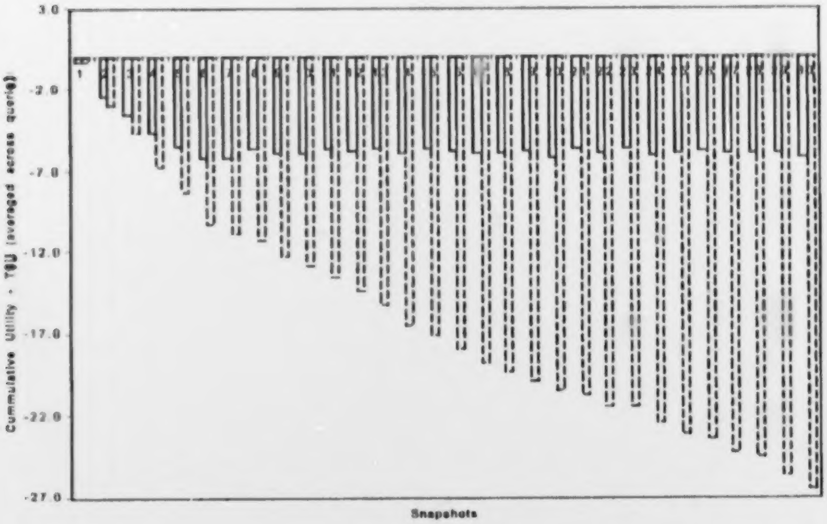


Figure 4: Assessment of Declaration Threshold. Solid bar: adaptive; dashed bar: non adaptive

MeSh Run:

Figure 6 shows the performance in terms of utility score for our MESH run. As mentioned before for this task we employed a rule-based entity recognizer which uses the MeSH hierarchy, an associated lexicon of synonyms and a supplementary list of concepts such as drug names. This run involved index vectors that were populated using only the entities extracted from the source text.

Entity-based performance on the MeSH subset proved to be quite intriguing. In 92 of the topics our system yielded the highest score - in some cases substantially higher than median performance.

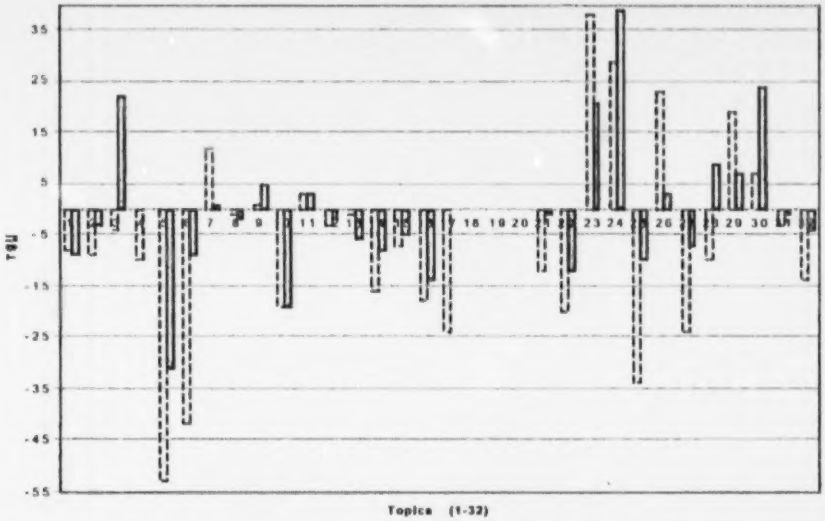


Figure 5: Assessment of Higher Break Threshold.
Dashed bar: 0.25 Solid bar: 0.3

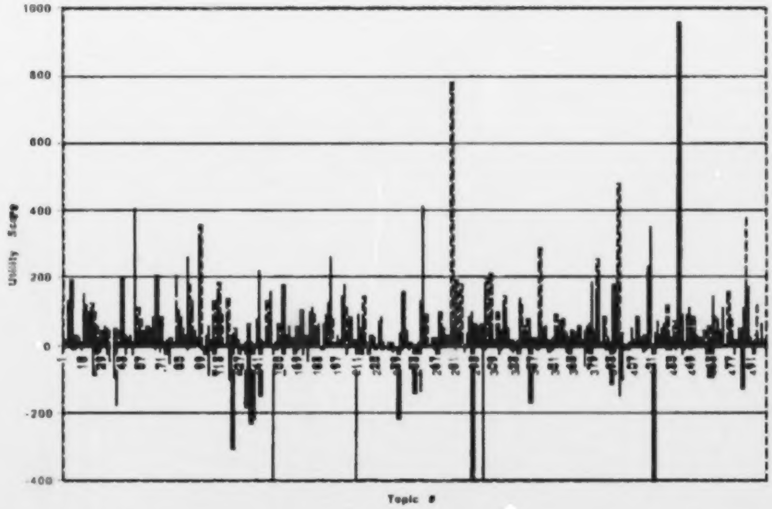


Figure 6: Performance of MESH Run. Solid line: median performance,
Dashed line: MESH run.

At the same time, in 147 of the topics our system yielded the lowest score - again in some cases substantially lower than median performance. We conjecture that the pure entity scoring yields high quality results - but for some topics our secondary cluster scheme is generating too many high-relevance clusters that prove to be off-topic. This may be due in part to the score being generated by ancestor/descendant MeSH term tree matches. Figure 7 presents performance (utility score) as

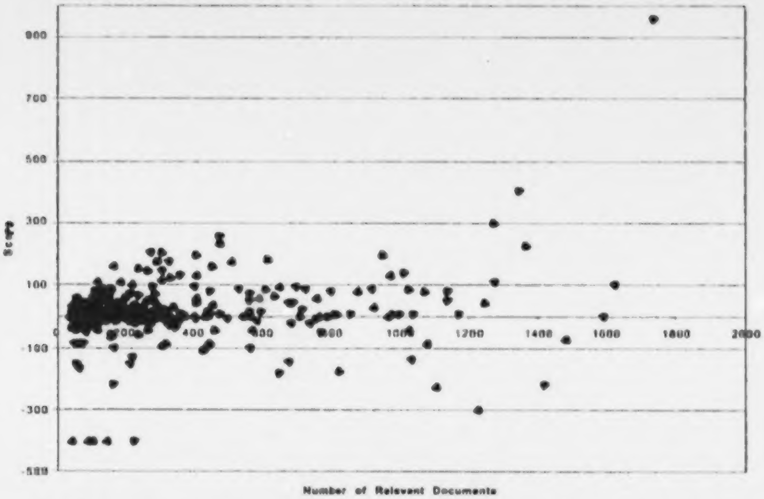


Figure 7: Performance versus Number of Relevant Documents for Topic

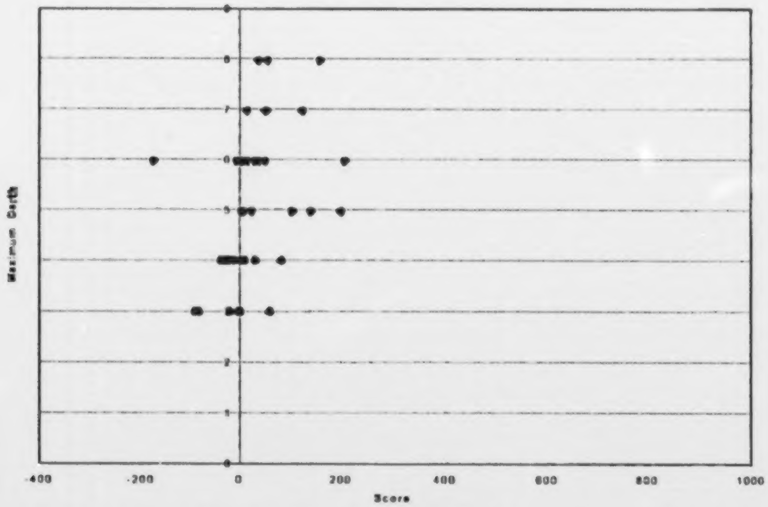


Figure 8: Maximum Depth of Topic's MeSH Phrases versus Performance

a function of the number of relevant documents present for a given topic. The figure shows 500 data points one for each topic. One may observe a general trend that as the availability of relevant documents improves, performance increases. Moreover, most of the scores are on the positive side of the Y axis. Figure 8 explores a different aspect. Our process extracts MeSH descriptors for each topic description from the MeSH hierarchy. In the figure we plot the maximum depth expressed by the group of extracted MeSH phrases for a topic and plot this against utility score. There are 50 data points corresponding to the first 50 MeSH topics. One may observe that except for a few outliers there is a slight trend for scores to improve with the ability to identify deeper i.e., more specific

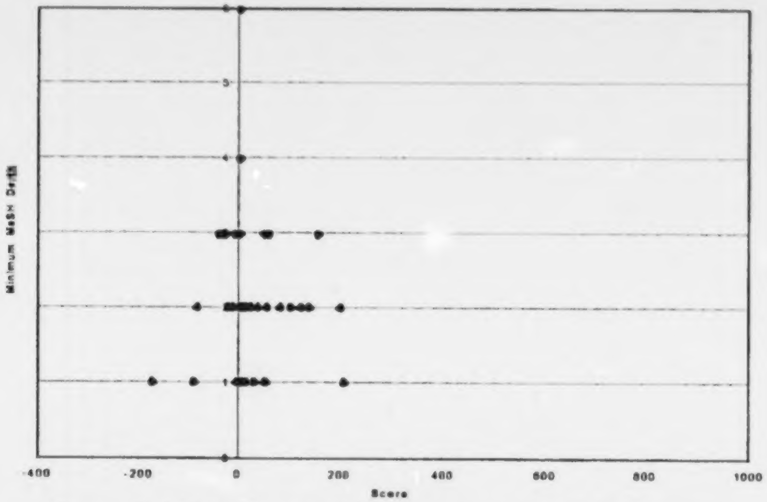


Figure 9: Minimum MeSH Depth for Topic versus Score

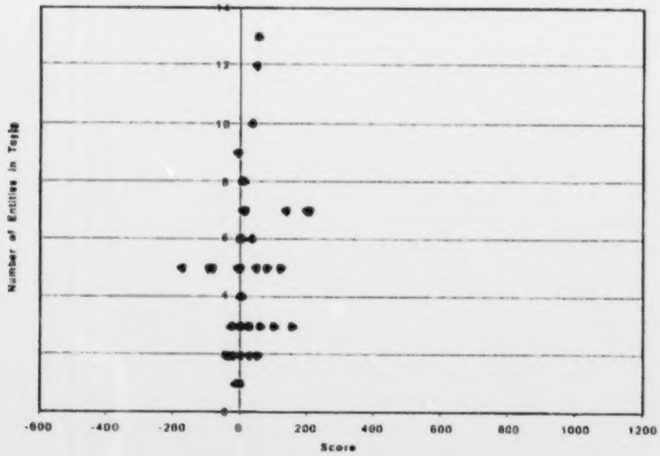


Figure 10: Number of Entities Recognized from Topic versus Performance

MeSH phrases. Interestingly the same sort of analysis using minimal MeSH depth for the topic, as shown in Figure 9, does not yield a recognizable trend. We also explored the effect of entity recognition on performance. Figure 10 represents the number of entities recognized on the Y axis and performance on the X axis. The graph shows that barring a few exceptions there appears to be a slight trend for performance to improve as the number of entities recognized increases.

In summary, the switch in domain from the newswire domain to MEDLINE proved to be challenging. The thresholds used in our submitted run were essentially our best guesses. For the future we also plan to explore different term weighting strategies as well as query expansion strategies prior to starting the filtering run.

2 – Question Answering:

We submitted two runs for this track, UIQA001 and UIQA002. Both utilized only the top 50 documents that were retrieved and distributed by Singhal. UIQA001 gave the better performance score with mean reciprocal rank of 0.227(strict) and 0.245 (lenient). The other run gave almost identical scores. Our overall QA approach is shown in below.

Document processing:

1. Extract only the textual parts of each document
2. Apply a sentence detection program to identify distinct sentences. We use the publicly available nxterminator program for this.
3. Apply part of speech tagging on each sentence
4. Apply our rule-based entity tagger on each sentence
5. Create a database of sentences formatted for the SMART retrieval system. Here each record corresponds to a single sentence with 3 different fields. The first holds the original untagged sentence, the second field holds the tagged sentence while the last field of the record holds only the particular entities extracted from each sentence.
6. Retrieve the top N sentences for each query. Maintaining the three distinct fields for each sentence record allows us to explore the relative merits of using different types of information for retrieving the sentence most likely to contain the answer. Using SMART allows us to explore different weighting schemes during retrieval.
7. Post process each of the N sentences to extract the top five 250-byte segments.

Query processing:

1. Apply part of speech tagging on each query
2. Apply our rule-based entity tagger on each query. Notice in the case of the query where possible its focus (a specific entity type) is identified in addition to all the entities contained in the query. This focus is utilized during the post processing step in 7 above.

The two runs differ very slightly in the post processing stage. Generally, this step includes cleanup of the sentences to remove any non informative strings, reduction of each sentence to a 250 byte string around the query focus (if known), removal of duplicate answer strings, and selection of the top 5 phrases. The difference between the two is in the extent to which cleanup of the sentences was done. As our results show, this did not influence performance in any way since the two runs yield almost identical results.

Error analysis indicates much room for improvement. Due to insufficient time, we were able to implement only very simplistic 250-byte segment selection strategies that proved to be a significant problem for our system. Secondly, our performance was limited by the availability of the answer within the top 50 document sets distributed. Again with less time pressures we should be able to explore the 1K datasets and also conduct our own retrieval runs for the top 1K or so documents. The results indicate that our approach managed to extract the answers for about 38 to 40% of the questions.

References

[1] Eichmann, D., M. E. Ruiz and P. Srinivasan, "Cluster-Based Filtering for Adaptive and Batch Tasks," *Seventh Conference on Text Retrieval,* NIST, Washington, D.C., November 11 - 13, 1998.

[2] Eichmann, D. and P. Srinivasan, "Filters, Webs and Answers: The University of Iowa TREC-8 Results," *Eighth Conference on Text Retrieval,* NIST, Washington, D.C., November, 1999.

TREC-9 Experiments at Maryland: Interactive CLIR

Douglas W. Oard,[*] Gina-Anne Levow,[†]and Clara I. Cabezas,[‡]
University of Maryland, College Park, MD, 20742

Abstract

The University of Maryland team participated in the TREC-9 Cross-Language Information Retrieval Track, with the goal of exploring evaluation paradigms for interactive cross-language retrieval. Participants were asked to examine gloss translations of highly ranked documents and make relevance judgments, and those judgments were used to produce a new ranked list in which documents assessed as relevant were promoted and those assessed as nonrelevant were demoted. No improvement over fully automatic ranking was found, which suggests that additional work on user interface design and evaluation metrics is required.

1 Introduction

The principal goal of our research on cross-language information retrieval (CLIR) is, of course, to build systems that are useful for some ultimate purpose. In the Text Retrieval Conferences (TREC), ad hoc retrieval tasks such as the CLIR track are designed to model the process of an individual user searching for one or more previously unseen documents on some topic. Although some applications such as document alerting require fully autonomous operation, we are particularly interested in interactive applications in which user and machine seek to synergistically exploit the strengths of each to search more effectively together than either could in isolation. Our principal goal in TREC-9 was to begin our exploration of this synergy in the context of CLIR.

Interactive retrieval can be roughly divided into three stages: query formulation, search, and browsing. In the context of CLIR, search has received the vast majority of the attention (e.g., at the TREC, NTCIR, and CLEF evaluations). There has also been some attention given to query formulation issues (e.g, user-assisted query translation), both in research systems and in deployed applications (c.f., http://messene.nmsu.edu/ursa/arctos). We are, however, aware of only two reported user studies that have explored issues related to interactive document selection by cross-language searchers. In one, Oard and Resnik adopted a classification paradigm to evaluate browsing effectiveness in cross-language applications,

---

[*] Human-Computer Interaction Laboratory, College of Information Studies and Institute for Advanced Computer Studies, oard@glue.umd.edu

[†] Institute for Advanced Computer Studies, gina@umiacs.umd.edu

[‡] Department of Linguistics, clarac@umiacs.umd.edu

finding that simple gloss (i.e., word-by-word) translations allowed users to outperform a Naive Bayes classifier [3]. In the other study, Ogden et al., evaluated a language-independent thumbnail representation in the TREC-7 interactive track, finding that the use of thumbnail representations alone resulted in even better recall at 20 documents than was achieved using English document titles [4]. The logical next step is to combine the best of both experiments, working directly on retrieval results as Ogden et al. did, while focusing on the marginal improvement over fully automatic processing as Oard and Resnik have done.

One obvious approach to this challenge would be to organize a TREC track at the intersection of the present CLIR and interactive tracks. We used the TREC-9 CLIR track for exploratory work in that direction, providing users with simple gloss translations of the retrieved documents and allowing them to improve the ranked list by moving documents that they believe to be relevant higher in the list and documents that they believe to be nonrelevant lower. Since either change would improve mean uninterpolated average precision if the user's judgment were correct, we adopted the change in mean uninterpolated average precision between the automatically generated ranked list and manually corrected ranked list as a metric for assessing the effect of the user's contribution.

2 Experiment Design

We initially conducted two small pilot studies to refine our user interface and experiment procedures using graduate students from our laboratory. Five graduate students from outside our laboratory with no self-reported Chinese language skills were then recruited as participants for the experiments reported below. All were proficient or native speakers of English, and all reported experience with document retrieval that was limited to the use of search engines such as AltaVista or Google. We offered to buy pizza for our participants upon completion of the experiment session, but all of them declined our offer!

The experiment was conducted during a single session. A Web-based user interface was designed specifically to support these experiments. The participants were first provided with an opportunity to become familiar with the system and the experiment protocol using two topics from the TREC-5/6 Chinese collection. For the experiment itself, we divided the 25 TREC-9 CLIR topics into sets of five, and assigned one set to each participant (i.e., topics CH55-CH59 to participant 1, topics CH60-CH64 to participant 2, etc.). The participants' task was to sequentially perform a search using a query that was automatically derived from the topic description and then judge the relevance of as many documents as time allowed based on th[illegible] understanding of the full topic description. This process involved four steps for each topic:

- Topic selection. Participants were instructed to click on the appropriate topic number from an initial selection page, resulting in display of the full topic description. After all participants completed reading the topic description, the participants were instructed to select the 'Search' button.
- Document selection. Selecting the search button resulted in addition of a ranked list of document titles to the same window. For each document in the list, a gloss translation

of the title and three radio buttons ('Relevant', 'Not relevant,' and 'No response') were presented. The participant was given five minutes from the time he/she hit 'search' to evaluate the relevance of as many documents as possible. We limited the displayed portion of the ranked list to fifty documents because that was far more than any participant in our pilot study could evaluate in 5 minutes. The participant could look at the translation of any document by clicking on the translated title. The first time this was done, a second window was created in which the translation was displayed. The document selection window remained visible in order to facilitate recording the relevance judgment and selecting the next document.

- Relevance judgment. If the participant was able to decide on the relevance of a document based on either the translated title or the translated document, he/she could select the 'Relevant' or 'Not Relevant' button for that document in the document selection window. The third option, 'No Response,' was initially automatically selected, and could be left selected if no judgment could be made.

- Recording relevance judgments. After five minutes, participants were instructed to manually select a button to submit their relevance judgments. The judgments were then recorded, and the topic selection screen was displayed to begin a new search.

We used a document translation strategy for CLIR, which is a natural choice when browsable translations must be immediately available. Our tools are designed to work with the GB character set, so we used the commercial NJStar Communicator package to convert from Big5 to GB. Fully automatic segmentation was then performed using the `ch_seg` package from New Mexico State University. We performed a term–by–term translation from Chinese into English using a balanced translation strategy to produce exactly two terms for each Chinese term in the original documents. For Chinese terms with no known translation, the untranslated Chinese term was converted to pinyin (without tone) and generated twice. For Chinese terms with one known translation, that translation was generated twice. Terms with two or more known translations resulted in generation of each of the "best" two translations once. The Brown Corpus served as a side collection to sort candidate translations in decreasing order of English usage (see [1] for additional details on this process). In prior experiments, we have found that such a balanced translation strategy significantly outperforms a more naive (unbalanced) technique in which all known translations are included because it avoids over-weighting terms that have many translations. The resulting English collection was then indexed using Inquery (version 3.1p1), with the default kstem stemmer and the default English stopword list.

We displayed the same 2-best balanced translations to the user. To improve readability, we grouped alternate translations using parentheses and showed the most common translation first using a bold font. Query terms were highlighted in red in an effort to help guide the user's eye to relevant passages. Our baseline for retrieval effectiveness was the mean uninterpolated average precision achieved by the automatically generated ranked list.[1] We used this as a basis for comparing three reranking approaches in our official run: Maximum, Partial, and Balanced.

---

[1] Our baseline run is unofficial, having been scored locally using the published relevance judgments.

In Maximum reranking,[2] documents marked by the participants as relevant were moved to the top of the list (position 1) and documents marked as irrelevant were moved to the bottom (position 1000), with the relative order between documents marked in the same way preserved. The remaining documents (labeled 'No Response') appeared in their original (automatically computed) order between those two sets. If the participant's relevance judgments were perfect (perfection here being defined by TREC assessors, of course), Maximum reranking would produce the greatest possible improvement in mean uninterpolated average precision. At least three possible sources of error are possible however:

- The participant might disagree with the TREC assessor's judgment of relevance, even if they fully understood the document.
- The participant might not be able to accurately assess the relevance of the document to the topic based on the gloss translation.
- The participant might select the wrong button by mistake.

Since moving a document all the way to the wrong end of the list could mask the beneficial effect on our metric of several correct assessments, we also tried two more conservative strategies. In Partial reranking,[3] documents marked as relevant were moved halfway to the top of the list. For example, a document in position 11 would be moved to position 6 if the participant marked it as relevant. Because of the way that uninterpolated average precision is computed, achieving a similar effect from demoting nonrelevant documents requires that the documents be moved further—we thus continued to move documents marked as 'not relevant' to the bottom of the ranked list (position 1000).

It is not clear how far down the list a document marked as nonrelevant should be moved, so we also tried a variant on Partial reranking that we called "Partial2".[4] As with Partial reranking, in Partial2 reranking we moved documents judged as relevant up by 50% of the distance to the top. When moving documents down, however, we limited their demotion to 10 times as far from the top of the list as they were in the automatically computed list. For example, a document in position 2 would move to position 11.

3 Results and Analysis

As shown in Table 1, we found that the best effectiveness of these four conditions was achieved by the Baseline (completely automatic) condition, although the differences were not statistically significant at $p < 0.05$ by a paired two-tailed t-test. We performed a query-by-query analysis to better understand this result and observed two important effects. First, as table 2 shows, when relevant documents are moved down the list, there can be a severe adverse impact on retrieval effectiveness, as these results on these two topics demonstrate. This suggests that we should adopt a more conservative strategy towards demotion.

---

[2] Official run 'TB.'

[3] Official run 'mixed.'

[4] Official run 'percent.'

	Baseline	Maximum	Partial	Partial2
All topics	0.2477	0.1710	0.1801	0.2183
Without CH60-CH64	0.1947	0.1803	0.1917	0.1916

Table 1: Official results and contrastive results with one participant removed.

Topic	Relevant Docs	Baseline	Maximum	Partial	Partial2
CH60	4	1	0.0031	0.0031	0.6429
CH62	1	0.5	0.0011	0.0011	0.025

Table 2: Degradation in average precision when the Baseline does well.

As it turns out, both of these topics were assigned to the same participant. On closer inspection, it is clear that our results seem to be adversely affected by a single participant. Each participant inspected the results of five queries and, due to time constraints, each query was inspected by only a single participant. As table 1 shows, when the topics presented to that participant (CH60-CH64) are excluded, virtually all of the differences are removed. We observed that the participant in question had judged two to three times as many retrieval results as other participants, and had marked the vast majority as not relevant, even when the title alone seemed to us to provide explicit evidence that the document was indeed on topic. These judgments are thus highly suspect, and in future studies it would clearly be desirable to assign the same topic to more than one participant [2].

We conducted some additional *post-hoc* analysis to find the optimum way of using the relevance judgments that we obtained. We grouped the relevance judgments into four categories:

TR Judged by the user as relevant based on the title

TN Judged by the user as not relevant based on the title

DR Judged by the user as relevant based on the document text

DN Judged by the user as not relevant based on the document text

For each category (and for the full set of user judgments), we computed the mean average precision for what we call Balanced reranking, using the following formula:

$$R' = \lfloor R(1-\Delta) \rfloor \tag{1}$$

$$R' = \lceil \frac{R}{1-\Delta} \rceil \tag{2}$$

where R' is the new rank, R is the original rank and Δ is a number between 0 and 1 that specified the increment size. Equation (1) is used for upward movement of documents judged to be relevant and equation (2) is used for downward movement of non-relevant documents.

We call this Balanced reranking because moving a document down by an increment of size Δ and then back up by an increment of size Δ would return it to its original position (except as influenced by roundoff errors). We tried every value for Δ between 0.0 and 1.0 in increments of 0.05.

Figure 1 shows the results of this *post hoc* analysis. None of the judgment subsets or values for Δ produced more than a 1% relative improvement in uninterpolated mean average precision. For higher values of Δ, it does appear that judgments based on examination of the full text of a glossed document were more reliable than judgments based on examination of the glossed title alone. When only titles were observed, the results suggest that decisions that a document was relevant may have been more reliable than decisions that a document was not relevant. Both results should be interpreted with caution, however. It is not possible to conclude that glossed documents are more informative than glossed titles, for example, because other factors (e.g., more careful participants) might explain the observed relationship equally well. Similarly, the relative effect of relevant and not-relevant judgments is sensitive to both the reliability of the judgments and the design of the Balanced reranking technique.

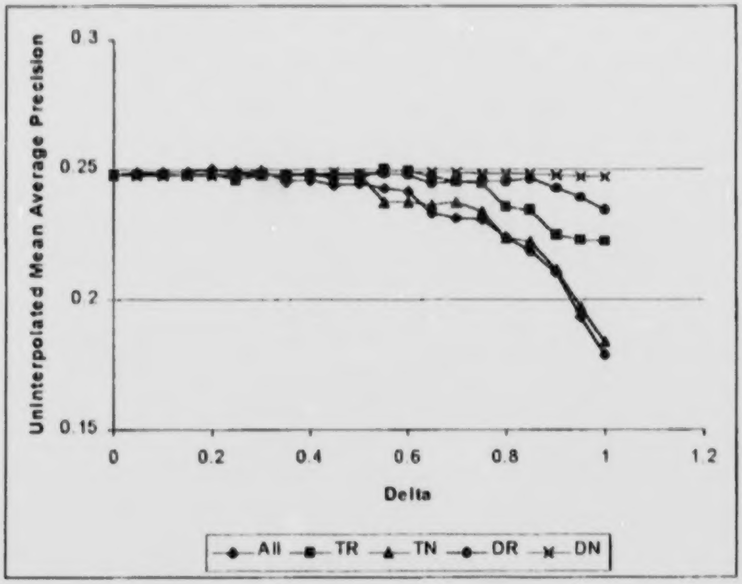


Figure 1: Balanced reranking with various subsets of the user judgments.

One important qualitative observation that we made is that our participants seemed to find the assessment process itself to be fairly difficult. NIST assessors are generally highly trained analysts, but our participants were (by design) novice users. If we were to provide more training before the study and more time to perform assessments, we might be able to minimize the effect of this factor.

4 Conclusion

We have tried to study interactive CLIR in the context of the present TREC CLIR track. The study reported here, with only a single participant for each block of five queries, a limit of five minutes to examine a full set of translations, and only a single interface design, is clearly of such limited scope that it would be difficult to draw any firm conclusions. Viewed as a pilot study for a more comprehensive interactive cross-language TREC evaluation, it offers some useful insights into the challenges of conducting such evaluations that we expect will inform our future work. Interactive reranking may ultimately have potential as a way of assessing user-system synergy, but clearly several issues of user training and study design remain to be worked out.

Acknowledgments

The authors are grateful to Bill Ogden for several helpful discussions about evaluation of interactive CLIR. This work was supported in part by DARPA contract N6600197C8540 and DARPA cooperative agreement N660010028910.

References

[1] Gina-Anne Levow and Douglas W. Oard. Translingual topic tracking with PRISE. In *Working Notes of the Third Topic Detection and Tracking Workshop*, February 2000.

[2] Douglas W. Oard, Gina-Anne Levow, and Clara I. Cabezas. CLEF experiments at Maryland: Statistical stemming and backoff translation. In Carol Peters, editor, *Proceedings of the First Cross-Language Evaluation Forum*. 2001. To appear. http://www.glue.umd.edu/~oard/research.html.

[3] Douglas W. Oard and Philip Resnik. Support for interactive document selection in cross-language information retrieval. *Information Processing and Management*, 35(3):363–379, July 1999.

[4] William Ogden, James Cowie, Mark Davis, Eugene Ludovik, Hugo Molina-Salgado, and Hyopil Shin. Getting information from documents you cannot read: An interactive cross-language text retrieval and summarization system. In *Joint ACM DL/SIGIR Workshop on Multilingual Information Discovery and Access*, August 1999. http://www.clis.umd.edu/conferences/midas.html.

INQUERY and TREC-9

James Allan, Margaret E. Connell, W. Bruce Croft,
Fang-Fang Feng, David Fisher, and Xioayan Li

Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, Massachusetts USA

This year the Center for Intelligent Information Retrieval (CIIR) at the University of Massachusetts participated in three of the tracks: the cross-language, question answering, and query tracks. We used approaches that were similar to those used in past years.

In the next section, we describe some of the basic processing that was applied across most of the tracks. We then describe the details for each of the tracks and in some cases present some modest analysis of the effectiveness of our results.

1 Tools and Techniques

Although UMass used a wide range of tools, from Unix shell scripts, to PC spreadsheets, three major tools and techniques were applied across almost all tracks: the Inquery search engine, query processing, and a a query expansion technique known as LCA. This section provides a brief overview of each of those so that the discussion does not have to repeated for each track.

1.1 Inquery

All three tracks used Inquery[Callan et al., 1992] as the search engine, sometimes for training, and always for generating the final ranked lists for the test. We used Inquery V3.2, an in-house development version of the Inquery system made available by the CIIR (V3.1). The differences between the two are not consequential for this study.

The current belief function used by Inquery to calculate the belief in term t within document d is:

$$w_{t,d} = 0.4 + 0.6 \times \frac{\mathrm{tf}_{t,d}}{\mathrm{tf}_{t,d} + 0.5 + 1.5\frac{\mathrm{length}(d)}{\mathrm{avg\ len}}} \times \frac{\log \frac{N+0.5}{n_t}}{\log N + 1}$$

where n_t is the number of documents containing term t, N is the number of documents in the collection, "avg len" is the average length (in words) of documents in the collection, length(d) is the length (in words) of document d, and $\mathrm{tf}_{t,d}$ is the number of times term t occurs in document d. The "tf" component is usually referred to as an "Okapi tf" function [Robertson et al., 1995], and the "idf" component is the normalized idf used by the CIIR for years.

1.2 Query Processing

The processing of queries—i.e., the transformation from TREC topic to InQuery query—was handled very similarly to the way it was managed for TREC-8, though there were some small changes. It includes three steps:

1. Basic Query Processing removes stop words and phrases (e.g., "relevant documents will include"), and stop structures—i.e., sentences discussing criteria of non-relevance in the narratives. For removing the stop structures the processor simply segments each sentence, then removes the sentence fragments that contain the stop structure (e.g., "Documents discussing ... are not relevant"), but keep those clauses on the negative part of the removed sentence (such as in "... not relevant, unless ...", where the "unless" clause should not be removed).

2. Query Formalizing identifies noun phrases (as in earlier TRECs), and proper names. The proper names were transformed to use the ordered proximity-one operator (e.g., `#passage25(#1(Golden Triangle)`, which requires that the two words occur immediately adjacent in the text in that order; the passage operator affects the weighting of the feature. For noun phrases we not only used the phrase operator but also duplicated the single terms used by the phrase (e.g., `#passage25(#phrase(tropical storms)), tropical, storms`). Note that for proper names, the single term duplication was *not* done.

   Query formalizing also identifies compound words, like *wildlife* and *airport*, that are formalized with the synonym operator (e.g., `#syn(#1(air port) airport)`). Also as part of query formalization, if there is any word concerned with foreign countries, like *international*, *world*, or *Europe*, a token `#foreigncountry` is added to the query. If there is a term concern with the United States, then a token `#usa` will be added. If both `#foreigncountry` and `#usa` are found in a query, all such tokens are removed.

   Finally a query is formed with the weighted sum operator (`#wsum`) with a weight for each term. For those terms occurring in the title and description fields the weight 1.0 is used, while 0.3 is used for those terms occurring in the narrative field. That is, we trust the title and the description more than the narrative.

3. Query Expansion adds 50 LCA concepts to each query. These 50 concepts are collected from top 30 passages (the passage database is built with TREC volumns 1 through 5). This is the same process that we used in past TRECs, but we did not use "filter-required" on the title words that has sometimes been used. LCA is described next.

1.3 Local Context Analysis (LCA)

In SIGIR 1996, the CIIR presented a query expansion technique that worked more reliably than previous "pseudo relevance feedback" methods.[Xu and Croft, 1996] That technique, Local Context Analysis (LCA), locates expansion terms in top-ranked passages, uses phrases as well as terms for expansion features, and weights the features in a way intended to boost the expected value of features that regularly occur near the query terms.

LCA has several parameters that affect its results. The first is the choice of LCA database: the collection from which the top ranked passages are extracted. This database could be the test collection itself, but is often another (perhaps larger) collection that it is hoped will broaden the set of likely expansion terms. In the discussion below, if the LCA database is not the test collection itself, we identify what collection was used.

LCA's other two parameters are the number of top passages used for expansion, and the number of expansion features added to the query. The LCA features were put into a query construct that allows a weighted average of the features. Assuming n features, f_1 through f_n, they are combined as:

BEST COPY AVAILABLE

$$
\begin{array}{lll}
\texttt{\#wsum(1.0} & 1.0 & f_1 \\
 & \vdots & \vdots \\
 & 1-(i-1)*0.9/s & f_i \\
 & \vdots & \vdots \\
 & 1-(n-1)0.9/s & f_n \;)
\end{array}
$$

Here, s is scaling factor that is usually equal to n. The weighted average of expansion features is combined with the original query as follows:

$$\texttt{\#wsum(1.0 } 1.0 \text{ original-query} \qquad w_{\mathrm{lca}} \text{ lca-wsum)}$$

where w_{lca} is the weight that the LCA features are given compared to the original query. Note that the final query is a weighted combination of the original query and the expansion features.

2 Cross-language Track

The TREC-9 cross-lingual track was done on three collections, *Hong Kong Commercial Daily*, *Hong Kong Daily News*, and *Ta Kong Pao*, totaling 127938 documents. All documents were encoded in BIG5 font. Since most Chinese tools work only for the GB font, the collection was converted into GB with the converter, hc3[1]. The database was built using Inquery's Chinese version of build (called "hanbuild") with each Chinese character indexed as a term.

In order to do the cross-lingual run, a machine translator is required to translate the English query into the target language, Chinese. A dictionary-based translation was used. The title and description fields of the topics were selected for constructing the queries in the source language, English. For each English query term (i.e., word or phrase), the translator consulted an English-Chinese dictionary, and replaced the English term with Chinese entries found in the dictionary. Using such a simple translator has weaknesses. The dictionary has a limited number of entries, so there may not be translations available for some English words or phrases. Also, there are multiple translations in Chinese for most English words, introducing ambiguities. An effort was made to both enlarge the dictionary and add query terms based on LCA to counteract the weaknesses.

2.1 Cross-language dictionary

The dictionary was built primarily from the English to Chinese dictionary obtained on-line from LDC.[2]. When experiments were run trying to translate the English version of the topics given in TREC-5 and TREC-6, this dictionary of 110,818 entries was found not to have translations of some critical English terms. In order to enlarge the dictionary, two Chinese-English dictionaries, one from the LDC[3] and one from elsewhere on the Web[4] were converted into English-Chinese dictionaries and merged with the original. With the merged dictionary performance on training queries improved.

The process of converting and merging the Chinese-English dictionary into English-Chinese was simple. Each English word or phrase given as a translation of the Chinese entry became an English entry with the original Chinese word as its translation. Explanations in parentheses or brackets were omitted. The conversion process merged identical converted English entries. For example, there are three Chinese entries concerning *nuclear power*,

---

[1] Available at http://www.cnd.org/software/UNIX.html.

[2] http://morph.ldc.upenn.edu/Projects/Chinese/

[3] http://morph.ldc.upenn.edu/Projects/Chinese/

[4] http://www.chinapage.com/dictionary/dictionary.html

BEST COPY AVAILABLE

```
核大国 /a nuclear power (country)/
核电   /nuclear power/
核能源 /nuclear power/
```

The converted English entries are identical, so they are merged as

```
nuclear power /核大国 /核电 /核能源 /
```

The converted English-Chinese dictionaries were merged with the original. If an entry already existed in the dictionary the new Chinese translations were just merged, otherwise a new English entry was added. After the merge, the resulting dictionary had 120,003 entries. This was the dictionary used in all training experiments described below.

Four smaller English-Chinese word lists were added to the dictionary. They were ITglossary[5] (1361 entries), Economy-Indus-glossary[6] (2037 entries), computer-terms[7] (602 entries), and sehk-stock[8] (2419 entries). After reformatting these dictionaries and merging them, the dictionary had 124,013 entries. This dictionary was used in the cross-lingual, TREC-9, final runs.

2.2 Cross-language query translation

The query translation was based on the English-Chinese dictionary lookup. The lookup procedure was a binary search on the alphabeticly sorted dictionary. The query text was segmented automatically with a list of stopwords. Each stopword indicated a new segment. A dictionary lookup was done for every sequence of English words within a segment (a potential phrase) from long sequence to short recursively. The translator first tried to find English phrases from the dictionary. If none were found, it translated single words. For example, in the description of query 1 of TREC-5 Chinese track, there is a sequence:

> most-favored nation status

The translator first looks in the dictionary for *most-favored nation status.* If it is not found, it then trys *most-favored nation.* If *most-favored nation* is found in the dictionary then the output is its Chinese translation otherwise it tries *nation status.* If a phrase lookup fails, the translator will do simple stemming of the plural head noun, in this case *most-favored nations* to *most-favored nation.* If no phrase is found in the dictionary, the translator translates single words. When a word-lookup fails the translator tries a stemmed or unhyphenated lookup. If all lookups fail the untranslatable term is discarded.

Name translations are important for queries about particular people, organizations, companies, or locations. There does not seem to be a dictionary covering proper names. An attempt was made to translate proper names using a parallel corpus. This approach was aborted because of time limitations.

When an English phrase or word was translated to a Chinese word consisting of more than one Chinese character, the multi-character word was represented by a proximity operator which forced the glyphs to be in order and adjacent. If more than one Chinese translation existed for an English phrase or word, all translations were wrapped in a synonym operator which treats all translations as the same word. While this avoids the exclusive use of wrong translations, it risks retrieving irrelevant material.

---

[5] http://www.iscs.nus.sg/ colips/archives/glossary/glossary.html

[6] http://news.cens.com/glossary/

[7] http://home.ust.hk/ lbsun/terms.html

[8] http://www.sehk.com.hk/index.asp?id=glossary.htm

2.3 Cross-language query expansion

Query expansion can be done either before translation, after translation, or both before and after translation.

Experiments were done expanding the English query prior to translation with LCA (described in Section 1.3). The expansion terms were chosen from TREC volumes 1-5.

Translated queries were also expanded using LCA. The translated query was used to extract expansion words from the top ranked passages of the original, TREC-9, Chinese database. The chosen passage size was 1500 words. Segmented words were used for expansion concepts. The segmenter was an improved version of the automatic segmenter used previously by the CIIR.[Ponte and Croft, 1996] It is based on hidden Markov models.

Stop phrases and phrases containing stopwords, digits and Chinese punctuation were automatically filtered from the expanded query.

Experiments were done using English topics from the TREC-5 and TREC-6 Chinese track. The results could be assessed using the available relevance judgements from this track.

The best results were found, using Chinese LCA only, selecting the top 20 passages and expanding the query with ten concepts. The expansion section of the query was weighted by 1.25 and each expansion concept was assigned a weight value $w(i) = 1.0 - 0.9(i-1)/70$ where i was the rank of the concept.

The experiments on TREC-6 Chinese LCA only were: a) Top 30 passages adding 5,10,15,20,30 ,40,and 50 terms. b) Top 20 passages adding 5,10,15,20,30 ,40,and 50 terms.

The experiments run on TREC-6 English LCA only were: a) Top 20 passages adding 5, 10, 15, 20, 30, 40, 50 terms. b) Top 30 passages adding 5, 10, 15, 20, 30, 40, 50 terms.

The experiments on TREC-6 with both English and Chinese LCA were: a) Top 20 passages in English, adding 10 terms. Top 20 passages in Chinese adding 5, 10, 15, 20, 30, 40, 50 terms. b) Top 30 passages in English, adding 10 terms. Top 30 passages in Chinese adding 5, 10, 15, 20, 30, 40, 50 terms.

2.4 Cross-language, monolingual contrast

In the monolingual run queries were processed in much the same way as was done for TREC-6. Queries were composed from the title and description fields of the topics. The queries in BIG5 font were converted to GB using the hc3 converter[9].

Segmentation was done at query time. The automatic segmenter was the same as that used for the cross-lingual run. Every segmented Chinese word was wrapped in a proximity operator, forcing its characters to be ordered and adjacent. When more than one word, represented by a single character, occurred in a sequence, the characters were enclosed in a phrase operator and restricted to be within 25 terms of each other. This was to correct for possible errors in the segmenter. Single isolated terms were down weighted by 0.3.

Stop phrases and terms were automatically filtered from the query.

Queries were expanded using LCA. The expansion concepts were extracted from the top 10 passages of the TREC-9 converted Chinese database of passages and concepts. The expansion was similar to the cross-lingual run, segmented words being used as concepts. The expansion part of the query was weighted by 2.0 and 40 weighted concepts were added. The weights of the concepts were the same as those in the cross-lingual run. The expansion words were automatically filtered to remove stop phrases.

Experiments were run on TREC-6 to find the best configuration of number of top passages, number of expansion concepts and weight of the expansion section of the query. The experiments were done with a weight of 2.0 and 2.5 for the expansion part, with the number of expansion terms at 10, 20,30,40, 50,60 and 70, and with the top 10,20 and 30 passages.

---

[9] http://www.cnd.org/software/UNIX.html

2.5 Cross-language conclusions

The submission consisted of four runs.

INQ7XL-1 was an official automatic run. It was cross-lingual using Chinese LCA only. The top twenty passages and ten expansion terms were used. The weight of the expansion part of the query was 1.25.

INQ7XL-2 was an official run. It was an official monolingual run using Chinese LCA. The top ten passages and forty expansion terms were used. The weight of the expansion part of the query was 2.0.

INQ7XL-3 was an optional run. It was cross-lingual using both English and Chinese LCA. The top 30 passages and 20 expansion terms were used for English and the top 20 passages and 10 expansion terms were used for Chinese. The weight of the expansion part of the query was 1.25

INQ7XL-4 was an optional run. It was cross-lingual using only English LCA. The top 30 passages and 20 expansion terms were used. The weight of the expansion part of the query was 1.25.

2.6 Cross-language results

The following table summarizes the cross-language results. We include the counts of comparisons with other systems because it gives a sense of where our approach worked well relative to others, and where it did not.

		AvgPrec comparison			
Run	AvgPrec	best	>avg	=avg	<avg
INQ7XL1	0.2416	3	12	0	10
INQ7XL2	0.2681	1	10	5	9
INQ7XL3	0.2425	2	17	1	5
INQ7XL4	0.2201	1	15	0	8

The table shows that all approaches were somewhat similar in their performance relative to other systems.

Not surpringly, INQ7XL2, the monolingual run, did the best overall, though it is not clear why that is true from the table (i.e., the distribution compared to other sites is different, but almost seems worse). Looking at other results shows that INQ7XL2 had a substantially higher precision at 20 and at 100 documents retrieved, meaning that staying within a single language did a better job at the high-precision end of the curve.

3 Question Answering Track

Our work in the question answering track is similar to the previous year. Numerous post hoc experiments indicated that performance on the TREC-8 long (250-byte) answer runs could have been improved by using a different query generator and a different passage selection method. This prompted us to submit two 250-byte runs to test if this result was an artifact of the nature of the TREC-8 question set, or generally applicable to question processing.

For the first run, INQ9AND, we generated initial queries from the questions by stripping the question words (who, what, where). We then added in the following expansions, which were derived from the TREC-8 questions:

1. a few synonyms, such as `#syn(length long)`, `#syn(far distance)`
2. a few alternations, such as `#syn(#uw5(between somewhere and here) #uw5(from here to somewhere))`
3. a proximity operator around superlative constructions, `#5(largest mountain)`

The resulting query was then wrapped in a probabilistic and (`#and`) operator. This approach resulted in roughly 10% higher performance on the TREC-8 question set.

The second run, INQ9WSUM, used the query generator from TREC-8, augmented with the the expansions listed above. These queries use the weighted sum (`#wsum`) operator, with all of the weights equal to 1.0 for the additional expansions. These augmentations produced a small (roughly 3%) improvement over the official INQ635 run from TREC-8.

For both runs, we used the LCA expansion of these queries to retrieve the top 20 documents. We then used the unexpanded query to retrieve the top passage from the top 5 documents from that set of 20. Based on our experiments with the TREC-8 questions, the unexpanded queries produce better answer passages than the LCA expanded queries. The passages are then reordered so that the rank order of the passages matches the rank order of the 20 documents retrieved with the LCA expanded query. The reordering step is performed because the LCA expanded queries tended to rank the answer containing documents higher than the unexpanded queries in our experiments with the TREC-8 question set.

Comparing the two runs, INQ9AND edges out INQWSUM on the aggregate score, 34.6% versus 33.6%. This is nowhere near the difference seen on the TREC-8 question set. Additionally, if we break down the scores by comparison against the median ranks for all of the systems, we find

	INQ9AND	INQ9WSUM
above	159 (23.3%)	171 (25.1%)
equal	370 (54.3%)	344 (50.4%)
below	153 (22.4%)	167 (24.5%)

that there is little difference between the two. INQ9WSUM produced 3 more answers than INQ9AND, and it produced 16 fewer rank 1 answers than INQ9AND (154 vs. 170). From this we conclude that there is some benefit from the changes based on the TREC-8 question set, but that the nature of the questions make them less than representative of real questions.

4 Query Track

The TREC-9 query track included 43 query files, variations of topics 51-100. There were three categories of the variations:

1. Very short: (2-3 words) based on topic.
2. Sentence: natural language, based on topic and judgements.
3. Manual Feedback: Manual NL sentence based on reading 5 or so relevant documents without reference to the topic (done by someone who doesn't have the topics memorized and who might use different vocabulary than the topic). This is an attempt to get a sentence which might use different vocabulary than the topic.

In this running of the track, we should get some feeling for whether query processing and expanding can improve the performance. So we did many experiments for different query processing techniques, and parameter settings for LCA query expansion.

4.1 Gathering queries

The queries were gathered from undergraduates in an information systems course at the University of Massachusetts. The information needed for query formulation was put on Web pages and the students were asked to formulate queries and submit them as part of a homework exercise. They received points for handing in validly formatted queries; there was no check to ensure that the quality of their queries were reasonable.

The students were randomly assigned two sets of five topics, based upon the last two digits of their student ID number (processing in advance determined that generated an even distribution over all the topics). The students were given these instructions:

> Each topic is listed with lots of descriptive information. In addition, there is a list of 5 or more documents that are known to be relevant to the topic. For each topics in set A of your batch do the following:
>
> - [short] Read the topic and come up with a short 2-3-word query that describes the topic as succinctly but completely as you can manage. You can think of this as queries that someone is likely to type on the Web, though that is not how they'll be used. Note that the `<title>` of the topic is not an example of a short query. An example of short query might be "popping corn methods."
> - [long] Now look at about 5 of the relevant documents for that same topic and generate a new and improved version of the query, this time coming up with a natural language, full sentence question or sentence that is on the topic specified, but also captures what it is about the relevant documents that makes them similar. Note that the relevant documents are known to be relevant, so there has to be something. An example might be, "What are the methods for popping corn that do not use a microwave?".
>
> Then for the five topics in set B of your batch, do the following:
>
> - [notopic] Look at about 5 of the relevant documents for the topic–without looking at the topic–and decide what it is about those documents that makes them all related to each other. Form a natural language, full-sentence statement or question that you think would be appropriate for retrieving those documents. For example, "How did people used to pop popcorn?" might be a query describing a bunch of documents that talk about ways to pop corn that seem to avoid talking about the microwave. Obviously there is a large amount of judgement here, so do not expect that everyone will have the same queries.

A total of 69 students provided queries in each of the three categories, meaning that 345 queries were provided in each category. Because some students failed to hand in the assignment, not all topics were evenly distributed. The end result was that the queries could be combined into five different sets of 50 queries for each of the different categories (long, short, notopic). The queries were combined somewhat randomly, though all five of any students queries were included in the same file. That means that any set of 50 queries includes queries generated by 10 different students.

Query sets INQ_a through INQ_e were created in Autumn 1998 and submitted for TREC-8 (in the same manner). Query sets INQ_f through INQ_j were created in Autumn 1999 and submitted for TREC-9.

Note that the generated queries were never checked for quality. It is theoretically possible that a student could submit nonsense queries. Some small problems of that nature happened and forced all participating sites to re-run their query track runs for some query sets.

4.2 Processing queries

For the query processing we ran experiments with four different query processors, the TREC-8 processor, the two QA processors described in the previous section, and another. The experiment results showed that the modified query processor performed better than all others.

The major difference between the modified one and the one used for TREC-8 concerns the handling of proper names. The TREC-8 query processor used to treat proper names differently from other phrases. When the TREC-8 processor identified a proper name, it used the unit proximity operater (i.e., #1) for a stronger constraint for the proper name occurrence. For the other phrases, it used the phrase operator (i.e., #phrase) and duplicated every single phrase term (i.e., terms occurring in the phrase). For example, in the topic 52, the country name *South Africa* was treated as a proper name, in the formatted query, it was

#passage25(#1(South Africa))

If *South Africa* was treated as other phrases, it would be

#passage25(#phrase(South Africa)) South Africa

The modification to that process loosens the constraint and treats all proper names as phrases. Overall experimental results from 5 query sets from TREC-8 (i.e., INQ3a, INQ3b, INQ3c, INQ3d, INQ3e) suggested this modification can improve the performace from 2% to 14%, with the average improved precision being 4.4%. However, this modification cannot guarantee improving the performance for every query, in which proper names occur. For example, the current processor can improve the topic 52 (containing *South Africa*) and 91 (containing *US Army*) with increased the average precision +83.5% and +860.2% respectively; but for the topics 69 (*SALT II*) and 70 (*Baby M*), the current processor makes the performance worse with decreased average precision -13.8% and -15.1% respectively.

There were two sets of experiments run for figuring out the best parameter setting for running LCA query expansion. One set was run with INQ3a and the other one was with INQ3d. In TREC-8 we did not run such experiments, we just borrowed the setting from the previous results of ad-hoc runs. The setting used in TREC-8 was selecting 50 expanded concepts from top 30 passages. For determining TREC-9 LCA setting we ran experiments trying from top 10 to 50 passages, and selected from 10 to 50 concepts for both query file INQ3a and INQ3d. The results showed that using top 50 passages and expanding 50 concepts can always improve the performance, the average precision was about 4% higher than using the TREC-8 settings (i.e., top 30 passages and 50 expanded terms).

We ran all 43 query sets that were part of the track using each of three different approaches: the query as is, basic query processing, and the addition of query expansion. More specifically:

- INQ7a — The queries were run by InQuery as is. InQuery stopped and stemmed the queries and treated them as if they were entered with InQuery's default `#sum` operator. That operator calculates the belief of every query term for a document, and then averages them.
- INQ7p — The queries were first passed through the query processing described above.
- INQ7e — After query processing, the queries were then expanded using LCA as described above.

For those who are curious: the INQ stands for InQuery, the name of the system used; the number 7 is an historical artifact of submission numbering from the start of the CIIR's participation in TREC; "a" and "p" and "e" should be self-explanatory.

4.3 Query track results

Overall, the three query processing approaches worked as expected. Query processing improved the quality of the queries, and expanding them improved the accuracy more. A brief summary over all 43 query sets shows:

Run	Rank	Below Average	Above Average	Top Rank	AvgPrec
INQ7a*	8	6	37	0	0.1799
INQ7p*	6	2	41	0	0.1848
INQ7e*	2	0	41	2	0.2288

Where "rank" respresents where (on average) the indicated run was in the set of 18 runs submitted. Note that the ranking is over query sets: so there were two query sets (of 43) that INQ7e did better than any of the other 18 runs.

4.4 Query track analysis

We did some more careful analysis of two of the queries to get a sense of how variant forms of the query failed or did not. In several parts of this analysis, we ran queries using the InQuery engine.

4.4.1 Analyzing Q51, Airbus subsidies

The original "query" was: *Document will discuss government assistance to Airbus Industrie, or mention a trade dispute between Airbus and a U.S. aircraft producer over the issue of subsidies.*

In looking at the various queries, there seemed to be four classes of terms that might be included in queries:

1. Key terms were terms without which the query did very poorly. For the query, any query without the term *airbus* did very poorly. The term as a query on its own yielded an average precision of 0.3364. Queries without that query tended to score about 0.01 average precision.
2. Supporting terms are those that are not useful on their own, but that improve results when they are included wiht a key term. For this query, the words *subsidy*, *Europe*, and *government* fell into this category. Some example queries and the effect on average precision:

Airbus	0.3364
Airbus *subsidy*	0.6029
Airbus *Europe*	0.4553
Airbus *government*	0.5543
Airbus *subsidy Europe government*	0.6483

3. Co-supporting terms help the query if supporting terms are present, but hurt the query if the supporting queries are missing. The word *industry* falls into that category here:

Airbus	0.3364
Airbus subsidy Europe government	0.6483
Airbus subsidy Europe government *industry*	0.6572
Airbus *industry*	0.3200

4. Noise terms are those that always hurt performance. For this query, *McDonnell* is an example of a noise term, even though on its surface it seems like it might be a useful term:

Airbus	0.3364
Airbus *McDonnell*	0.2253
Airbus subsidy Europe government	0.6483
Airbus subsidy Europe government *McDonnell Douglas*	0.4104

We also found several terms that affected the quality of the result, ranging from misspellings (e.g., "Concortium" or "Mcdonnel") to unusual stemming operations ("Boeing" to "boee"). The spelling errors clearly caused problems; the stemming areas are more amusing than anything else, since both queries are documents were stemmed with the same techniques. (Ironically, misspelling "McDonnell" turns out to be a win since "McDonnell" is a noise term.)

4.4.2 Analyzing Q93, NRA backing

The original "query" was: *What backing does the National Rifle Association have?*

We analyzed this query looking at the content of the query: what topics were mentioned and which were necessary. We found:

- Almost all queries had either *NRA* or *National Rifle Association* in them. The former was substantially less effective than the latter.
- Queries that included *gun* did well, but *handgun* (without *gun* also) performed terribly. When *gun* was combined with *National Rifle Association* the queries did very well.
- Words related to "support" or "backing" were important but there was no obvious pattern to what worked or did not.
- Additional "filler" words (e.g., "congress" or "people") had a range of effects, and those effects varied across systems. That is, "people" might hurt the query on one system, but have no effect on another.

For this query, we also looked at the effect of query expansion on the queries. We found that expanding queries with *NRA* made them as good as the original *National Rifle Association* queries, but that that latter set of queries doubled or tripled in effectiveness when expanded! The query expansion also removed the system differences do to handling of filler words.

5 Conclusions

We participated in three tracks this year. In the cross language track, we experimented some techniques for crossing the character encoding boundaries. Our efforts were moderately successful, but we do not believe that our approach worked well in comparison to other techniques.

In the question answering track, we focused on bringing answer-containing documents to the top of the ranked list. This is an important sub-task for most methods of tackling Q&A, and we are pleased with our results. We are now looking at alternate ways of thinking about that task that leverage the differences between retrieval for Q&A and for IR.

Finally, we continued to participate in the query track, providing large numbers of query variants, and running our system on the huge number of resulting queries. Our analysis showed how query expansion compensates for some of the problems that can occurs in query formulation.

6 Acknowledgements

This work was supported in part by the National Science Foundation, Library of Congress, and Department of Commerce under cooperative agreement number EEC-9209623, in part by the Air Force Office of Scientific Research under grant number F49620-99-1-0138, and in part by SPAWARSYSCEN-SD grant number N66001-99-1-8912. Any opinions, views, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsors.

References

[Callan et al., 1992] Callan, J. P., Croft, W. B., and Harding, S. M. (1992). The INQUERY retrieval system. In *Proceedings of the Third International Conference on Database and Expert Systems Applications*, pages 78–83, Valencia, Spain. Springer-Verlag.

[Ponte and Croft, 1996] Ponte, J. and Croft, W. B. (1996). Useg: A retargetable word segmentation procedure for information retrieval. In *Proceedings of the Symposium on Document Analysis and Information Retrieval (SDAIR).*

[Robertson et al., 1995] Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M., and Gatford, M. (1995). Okapi at TREC-3. In Harman, D. K., editor, *The Third Text REtrieval Conference (TREC-3).* NIST.

[Xu and Croft, 1996] Xu, J. and Croft, W. B. (1996). Query expansion using local and global document analysis. In *Proceedings of the 19th annual international ACM SIGIR conference on research and development in information retrieval*, pages 4–11, Zurich. Association for Computing Machinery.

Incrementality, Half-life, and Threshold Optimization for Adaptive Document Filtering

Avi Arampatzis[*] **Jean Beney**[†] **C.H.A. Koster** **T.P. van der Weide**

Proceedings of the Ninth Text REtrieval Conference (TREC-9),
Gaithersburg, Maryland, November 13–16, 2001.
Department of Commerce, National Institute of Standards and Technology (NIST).

1 Introduction

This paper describes the participation by researchers from KUN (the Computing Science Department of the Katholieke Universiteit Nijmegen, The Netherlands) in the TREC-9 Filtering Track. As first-time TREC participants, our group participated in all three subtasks — adaptive, batch, and routing — while concentrating mainly on adaptive tasks. We have made use of two different systems:

- FILTERIT, for the adaptive and batch-adaptive[1] tasks: a pure adaptive filtering system developed in the context of our TREC-9 participation. It is based on the Rocchio algorithm.
- LCS, for the routing and batch filtering tasks: a multi-classification system based on the Winnow algorithm.

In adaptive filtering, our contribution has been threefold. Firstly, we have investigated the value of retrieved documents as training examples in relation to their time of retrieval. For this purpose we have introduced the notion of the half-life of a training document. Secondly, we have introduced a novel statistical threshold selection technique for optimizing linear utility functions. The method can be also applied for optimizing other effectiveness measures as well, however, the resulting equation may have to be solved numerically. Thirdly and most importantly for adaptive long-term tasks, we have developed a system that allows incremental adaptivity. We have tried to minimize the computational and memory requirements of our system without sacrificing its accuracy. In the batch and routing tasks, we have experimented with the use of the Winnow algorithm, including a couple of small improvements.

From the two topic-sets given, we have experimented only with the 63 OHSUMED queries. We did not submit any runs on the 4904 MeSH topics; these were simply too many to be processed by our present systems in a reasonable time and space. All experiments were done using a keyword-based representation of documents and queries, with traditional stemming and stoplisting, although our long-term intention is to use phrase representations [2], and apply more sophisticated term selection methods [3]. Table 1 summarizes our official TREC-9 runs.

Next, we will briefly describe the pre-processing applied to the data. The FILTERIT and LCS systems are described in Sections 3 and 4, respectively. In Section 5 we give an overall view to how our systems performed in relation to other participants.

2 Stream pre-processing

We used only the title and abstract fields (.T and .W tags) of the OHSUMED documents; their MeSH-headings were discarded. The pre-processing of the documents and topics was minimal and quite traditional. It consisted of (in the order of application): replacement of all non-letters by spaces, deletion of all one-letter words, lowercasing, stoplisting[2], stemming[3], deletion of all one-letter stems, and DF-stoplisting (removal of the top-100 stems with the highest document frequencies in `ohsumed.87`).

In summary, our pre-processing was quick-and-dirty. There was no special treatment of proper names, all numbers were lost, and we made no use of multi-word terms such as phrases or word clusters. Moreover, we used no external resources such as online dictionaries or thesauri.

---

[*] Computing Science Institute, Faculty of Mathematics and Informatics, University of Nijmegen, Postbus 9010, 6500 GL Nijmegen, The Netherlands,
tel: +31 6 51 408838, fax: +31 24 3553450,
e-mail: `avgerino@cs.kun.nl`, `http://www.cs.kun.nl/~avgerino`

[†] On sabbatical leave from INSA de Lyon.

[1] We see the batch-adaptive task as an adaptive rather than a batch filtering task.

[2] We used the standard stoplist of the SMART system, `english.stop`, available from:
`ftp://ftp.cs.cornell.edu/pub/smart/`

[3] We used the Porter stemmer of the `Lingua::Stem` (version 0.30) library extension to PERL.

Task	Topics	Optimized for	System	Run-tag
adaptive	OHSUMED	T9U	FilterIt	KUNa1T9U
adaptive	OHSUMED	T9U	FilterIt	KUNa2T9U
adaptive	OHSUMED	T9P	FilterIt	KUNa1T9P
adaptive	OHSUMED	T9P	FilterIt	KUNa2T9P
batch-adaptive	OHSUMED	T9U	FilterIt	KUNbaT9U
batch	OHSUMED	T9U	LCS	KUNb
routing	OHSUMED	—	LCS	KUNr1
routing	OHSUMED	—	LCS	KUNr2

Table 1: TREC-9 filtering runs submitted by KUN.

3 The FilterIt System

The FilterIt system, which we used for performing the adaptive and batch-adaptive tasks, has been developed in the context of our TREC-9 participation. It is a pure adaptive filtering system based on Rocchio's method [10]. Rocchio's method performs well in a situation where only a few training documents are available, see e.g. [9], and this is exactly the case in the adaptive task. In such a situation, the initial query becomes important and the method can moreover deal in a suitable way with the topic descriptions.

We have modified the formula traditionally used for relevance feedback in order to allow for weighing of training documents according to their time-stamps. Moreover, the implementation of the algorithm we will present, allows very accurate incremental training of classifiers, without using any document buffers, so its memory and computational power requirements are low. In order to limit further the memory requirements of our system per topic, we also use a form of on-the-fly term selection.

Our system adapts queries and thresholds independently for each topic, meaning that the filtering model for a topic is updated after the retrieval of every single document for that topic. In the runs optimized for the T9P measure, threshold adaptations are even triggered independently of document retrievals.

For optimizing the filtering thresholds, we have introduced a new statistical technique which takes into account the relative density of relevant to non-relevant documents seen in the stream, and their score distributions. Most of the quantities that our technique requires can be updated incrementally, but a small document buffer seems unavoidable.

3.1 Incremental Query Training

The version of Rocchio's method traditionally used for relevance feedback is

$$Q = \alpha\, Q_0 + \beta \frac{1}{|\mathcal{R}|} \sum_{D \in \mathcal{R}} D - \gamma \frac{1}{|\mathcal{N}|} \sum_{D \in \mathcal{N}} D\,, \tag{1}$$

where Q_0 the initial query, $\mathcal{R}$ and $\mathcal{N}$ the sets of relevant and non-relevant documents respectively, and $|.|$ denotes the number of elements in a set. The parameters α, β, and γ control the relative contribution of the initial query, and that of the relevant and non-relevant documents to the new query Q. All components which end up with negative weights in Q are removed.

The initial query and the documents are usually represented by vectors weighted in a *tf.idf* fashion[4]. While the *tf* components are usually independent of corpus statistics, the *idf* components depend on the collection. Since in filtering the whole collection is not available in advance, the *idf* components should be updated over time (*incremental idf*). Therefore, it would be more suitable for filtering to keep these quantities separately. As a result, queries and documents in our system are only *tf*-weighted, e.g., a document D_i is represented by

$$D_i = [tf_{i1}, \ldots, tf_{iK}]\,, \tag{2}$$

where K the total number of terms known by the system at one point in time. Any document or query is a sparse array since it contains far less non-zero components than K, so they are implemented by *hash arrays*.

Since all vectors are only *tf*-weighted, we have moved the impact of *idfs* into the similarity function, which for a query Q and a document D has been defined as:

$$S(Q, D) = Q\; IDF\, D^T\,, \tag{3}$$

where IDF is the diagonal matrix $diag(idf_1, \ldots, idf_K)$, and X^T denotes the transposed array of X. Such an implementation allows, at any time, the usage of the latest *idf* values.

---

[4] *tf.idf* denotes here the *family* of weighting schemes which weigh a term proportionally to its frequency in a document or query and inversely proportional to its frequency across the collection. In practice, *tf* and *idf* are implemented by some monotonically increasing functions of the corresponding frequencies. We consider the decision to use a *tf.idf*-type weighting scheme as an *architectural* choice, while the exact form of the functions is an *implementational* choice. We give our implementational choice in Section 3.3.

Now, formula 1 can be calculated incrementally by simply re-writing it as

$$Q_n = \alpha Q_0 + \beta \frac{1}{R_n} B_n - \gamma \frac{1}{N_n} C_n \;, \qquad (4)$$

where B_n, C_n are the accumulated sums of the term frequency vectors of relevant and non-relevant documents respectively, and R_n, N_n are the numbers of documents in each category[5]. When document D_n is retrieved, Q_n is calculated in two steps. First, all time-dependent quantities (everything on the right side of the formula which has the subscript n) in the last formulation are updated. Then, the query Q_n is calculated using the updated quantities.

Summarizing, the architecture we have just described allows the most accurate incremental training with Rocchio. No training documents have to be discarded, as would have been necessary in a *sliding window* adaptive system. Moreover, no document buffers are necessary, except B_n and C_n in which all training documents are accumulated. In order to achieve all these, the only requirement is that *tfs* are *static* in the sense that they can be calculated only once when a document arrives.

Of course, there is another minor concession we make here, that is to allow counting registers of infinite width (the values of the components of B_n, C_n, and the variables R_n, N_n can grow up to infinity). Double precision arithmetic approximates this assumption well. In any case, when a number approaches the maximum width, all quantities can be divided by a constant without invalidating the model.

3.2 Convergence, Responsiveness, and Decay

The goal of the incremental training we have described so far is to gradually converge to a perfect classifier. All training documents, irrespective of their time of retrieval, are taken into account with equal importance in constructing the classifier. Systems that implement this kind of converging adaptivity we shall call *asymptotically adaptive*. The use of an asymptotically adaptive system for filtering implicitly assumes that topics are *stable*, i.e. there are no *topic drifts*.

If there are topic drifts, the position of the perfect classifier moves in the document-space. Therefore, it is beneficial for a filtering system to be capable of tracking a topic rather than converging. This capability can be achieved by weighing more heavily training documents that are retrieved recently. We call such systems *locally adaptive*. The choice between local adaptivity and asymptotic adaptivity should be made depending on whether *convergence* or *responsiveness* is more important. More about various forms of adaptivity for filtering systems and the nature of topics in filtering can be found in [1].

In TREC-9, topics are assumed to be stable, suggesting that an asymptotic behaviour would be more proper. However, the OHSUMED collection consists of documents collected in a period of five years and it is likely that for a topic the content of its relevant documents changes over the years, e.g., think of new treatments developed for the same sickness. The effect of such *document content drifts* is equivalent to *user interest drifts* in the sense that the idea of *relevance* changes.

In order to weigh training documents differently, we replace the average vectors in the Rocchio formula of Eq. 1 with *weighted averages*. This does not invalidate the motivation of the formula. For instance, the average vector of relevant documents becomes

$$\frac{1}{|\mathcal{R}|} \sum_{D \in \mathcal{R}} D = \frac{1}{\sum_{i:D_i \in \mathcal{R}} l_i} \sum_{i:D_i \in \mathcal{R}} l_i D_i \;, \qquad (5)$$

where l_i represents the weight with which the document D_i contributes to the average.

A heavier weighting of recently retrieved training documents may be implemented by a *decay* operation with *half life* h, i.e. the age that a document must be before it is half as influential as a fresh one in updating the query. If a document D_i is retrieved at time t_i, and the current time is t_n, we set

$$l_i = 0.5^{(t_n - t_i)/h} \;, \qquad (6)$$

where t_n, t_i, and h are measured in the same units, e.g., months.

Whether the initial query Q_0 should decay or not depends on the nature of a topic. For a drifting user interest, Q_0 should decay. For a stable interest with document content drifts (as we argued to be true for TREC-9), any of the two choices can be motivated (it rather depends on how Q_0 is formulated). For our official TREC-9 runs, we chose to decay Q_0.

The decay operation can be performed incrementally. When D_n is retrieved, and assuming that it is found to be relevant, then it is easy to show that average vectors, e.g., the one of relevant documents, can be updated as:

$$\frac{1}{R_n} B_n = \frac{1}{l R_{n-1} + 1} \left(l B_{n-1} + D_n \right) \;, \quad l = 0.5^{(g/h)} \;,$$

where $g = t_n - t_{n-1}$ stands for the elapsed time since the previous query update (i.e., since Q_{n-1} was calculated). Therefore, when a document is retrieved, all

---

[5] The convention we use for the subscript n is: n is the total number of training documents available (relevant and non-relevant). Training documents are the ones given at the time of bootstrapping (as for the batch-adaptive task), and all retrieved ones during filtering since their relevance judgment can be seen. Thus, Q_n is the classifier built using n training documents. If r of them are relevant, then $R_n = r$ and $N_n = n - r$, and B_n, C_n contain the sum of r and $n - r$ document vectors respectively.

BEST COPY AVAILABLE

time-dependent quantities of equation 4 are multiplied by the current decay factor l before they are updated with the new document. To maintain correct decaying weights, even the quantities which are not going to be updated have to be multiplied, e.g. even if D_n is relevant, $N_n = lN_{n-1}$ and $C_n = lC_{n-1}$.

In TREC-9, time is estimated on the number of documents seen in the stream. It is given that the stream produces, on average, around 6,000 documents per month. Therefore, for a half life of m months, we set $h = 6,000m$, and g is simply the number of documents filtered since the previous query update.

3.3 Term Weighting

For term weighting, we "borrowed" the *Ltu* formula from [11]. In the *Ltu* weighting scheme, L is the term frequency factor, t is the inverted document frequency factor, and u is the length normalization of the document or query.

The t factors were initialized from `ohsumed.87`. Then we used *incremental idf*: upon the arrival of a new document and before any other calculation is performed, all quantities that contribute to the t factors are updated

The application of the *Ltu* formula in adaptive filtering presents a small problem. The average number of unique terms per document changes over time, therefore, the term weights of past documents should be re-calculated as well. We chose to calculate this average document length on `ohsumed.87` and assume that it will not change in the future. This allows to calculate the u factor once and for all, when a document arrives. The assumption that the average document length will remain the same in the future is not far from reality for the OHSUMED collection, since there is no special reason why medical researchers should write abstracts of different lengths over time.

Summarizing and using our notation, the exact form of the term weighting we used is:

$$tf = L \times u' , \quad idf = t , \tag{7}$$

where u' is the same as u but with the average document length fixed on its `ohsumed.87` value (that was 40.8 after the pre-processing). This form presents *static tf* components, in the sense that they are calculated once when a document arrives without the need to re-calculate them in the future, and *dynamic idfs*. These features allow for incremental training, as we have shown in Section 3.1.

3.4 On-the-fly Term Selection

It is empirically known that as the size of a corpus grows, the number of unique words seen grows with the square-root of the number of documents. In case of multi-word terms (phrases), the number of such enriched terms grows even faster. Therefore, the number of components of B_n and C_n vectors grows, at least, with the square-root of the number of retrieved documents n. To limit the size of these vectors we use term selection.

In fact, term selection is more critical for the threshold optimization technique we will describe in Section 3.5. The incremental application of the optimization technique requires matrices as large as the *square* of the size of B_n or C_n, consequently the memory requirements may explode soon if no term selection is used (see Section 3.5.3).

Term selection was applied for each topic independently, before every incremental update of the corresponding query. Our *on-the-fly* term selection consists of the following steps. First, a query is constructed using information only from relevant instances and the current *IDF* matrix:

$$Q_{n,\mathrm{rel}} = \left(\alpha \, Q_0 + \beta \frac{1}{R_n} B_n \right) IDF \, . \tag{8}$$

Then, we rank all terms of $Q_{n,\mathrm{rel}}$ according to their weight, and select only the top-k ones and the terms occurring in Q_0. The rest of the terms are discarded and removed from all quantities kept by the system for the topic (e.g., B_n and C_n). Then, Q_n is calculated using the reduced data.

This technique limits the memory required for filtering a topic. However, the size of the *IDF* matrix still grows by the time, as previously unseen terms occur in documents of the stream. We consider *IDF* as *stream data* rather than *topic data*, since it is the same for all topics being filtered at any point in time. Therefore, we do not limit its size.

3.5 The Score-Distributional Threshold Optimization

Let us assume that for some topic a training stream of n documents is available, of which r are relevant. After the filtering the stream with some query and a threshold, each document may be classified under one of the four categories shown in the contingency table:

	relevant	non-relevant
retrieved	R_+	N_+
non-retrieved	R_-	N_-
total	r	$n-r$

The variables R_+, N_+, R_-, and N_- refer to the number of documents in each category. Given any evaluation measure M and a query, a filtering threshold θ can be selected so as to optimize the measure on the training stream. In this Section, we outline a threshold optimization technique which can be applied for any evaluation measure of the form $M(R_+, N_+, R_-, N_-)$, i.e. M is any function of the document counts in each category.

The idea is to describe a dataset of document scores with their probability density function. There are two such functions, one for relevant and one for non-relevant document scores. Then, these functions are multiplied by the corresponding numbers of document scores used for their estimation, so that the area below each curve amounts to the number of documents. Now, each variable of the contigency table can be expressed as a function of θ by integrating a range on the corrensponding curve, e.g. $R_+(\theta) = \int_\theta^{+\infty} r P_r(x)\,dx$ where P_r is the probability density of relevant document scores. The threshold which optimizes M is a solution of

$$\frac{dM\left(R_+(\theta), N_+(\theta), R_-(\theta), N_-(\theta)\right)}{d\theta} = 0\,. \qquad (9)$$

Depending on the exact form of function M, equation 9 may not have analytical solutions and it should be solved numerically.

3.5.1 Score Distributions

In [4] we prove that a Gaussian limit appears for the distribution P_r of relevant document scores. Furthermore, we show that the distribution approaches the Gaussian quickly, such that corrections go to zero as $1/K_Q$, where K_Q the length of the query. Empirically, Gaussian shapes form at around $K_Q = 250$.

For non-relevant documents, we show in the same study that a Gaussian limit is not likely, and if it appears, then only at a very slow rate with K_Q. Empirically, we have never seen Gaussian shapes even for all dimensions resulted from massive expansion of queries. Our empirical data, however, point out that the right tail of the distribution can be very well approximated with an exponential.

Figure 1 shows the empirical score distributions for TREC topic 352 on the Financial Times collection. We collected these data as follows. First, we trained a classifier using all relevant documents and an equal number of the top-scoring non-relevant using the *query zone*[6]. Then, we calculated the scores of relevant and non-relevant documents for the classifier. The middle plot shows the empirical distribution of relevant document scores, and the corresponding Gaussian multiplied by the number of scores. The left plot shows the empirical distribution of the top-100 non-relevant scores, and exponential curves of the form $c_1 e^{-c_2 x}$ fitted on the top 100, 50, 25, and 10 scores. It seems that at least 50 or more scores are needed for an accurate threshold estimate. The right plot shows the optimal T9U threshold. As we will prove in Section 3.5.2, the optimal T9U threshold is the intersection of the probability densities of relevant and non-relevant document scores, weighted as $2r$ and $n - r$ respectively.

3.5.2 Optimizing Linear Utility Functions

Let U any linear utility function of the form

$$U_{(\lambda_1,\lambda_2,\lambda_3,\lambda_4)} = \lambda_1 R_+ + \lambda_2 N_+ + \lambda_3 R_- + \lambda_4 N_-\,, \qquad (10)$$

where $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ denote the gain or cost associated with each document that falls in the corresponding category. The linearity of such measures allows for analytical solutions of Eq. 9, since the integrals describing the document counts cancel out with the derivative of the measure. Consequently, and after a few calculations, Eq. 9 becomes

$$\lambda\rho P_r(\theta) = P_{nr}(\theta), \quad \lambda = \frac{\lambda_3 - \lambda_1}{\lambda_2 - \lambda_4}\,, \quad \rho = \frac{r}{n - r}\,. \qquad (11)$$

ρ is the *relative density* of relevant to the non-relevant documents in the training stream. For T9U, $\lambda = 2$. P_r and P_{nr} are the density functions of the probability distributions of relevant and non-relevant document scores.

Let P_r be a Gaussian density with mean μ_r and standard deviation σ_r, and P_{nr} an exponential density of the form $c_1 e^{-c_2 x}$ estimated on the right tail of the distribution of the non-relevant scores. Then, the solution of Eq. 11, i.e. the optimal threshold, is:

$$\theta = \begin{cases} (b - \sqrt{\Delta})/a & \text{if } \Delta \geq 0 \\ +\infty & \text{if } \Delta < 0 \end{cases}\,, \quad \Delta = b^2 - ac\,,$$

$$a = \frac{1}{\sigma_r^2}\,,\ \ b = \frac{\mu_r}{\sigma_r^2} + c_2\,,\ \ c = \frac{\mu_r^2}{\sigma_r^2} - 2\ln\left(\frac{\lambda\rho}{c_1\sqrt{2\pi\sigma_r^2}}\right). \qquad (12)$$

Note that since the exponential corresponds to the top non-relevant scores, it does not extend accurately to low scores. Consequently, the method gives more accurate results when there is no contribution of N_- into the utility score, i.e. for $\lambda_4 = 0$, which holds for T9U.

3.5.3 Incremental Mean and Deviation

The method we have described for threshold optimization uses the mean of the relevant document scores and their standard deviation. In general, means and deviations can be calculated incrementally. However, in the case of filtering every update of the query causes the scores of the previous training documents to change. The choice that we have made in Section 3.1 to have query and documents only *tf*-weighted and keep *idf*s separately, allows for incrementality also here.

For query Q_n, the average score of r relevant documents $D_1, \ldots, D_r$ with scores $s_1, \ldots, s_r$ can be written as

$$\mu_r = \frac{1}{r}(s_1 + \cdots + s_r)$$

$$= \frac{1}{r}\left(Q_n\, IDF\, D_1^T + \cdots + Q_n\, IDF\, D_r^T\right)$$

---

[6] For the query-zoning method, see Section 3.6.2 or [12].

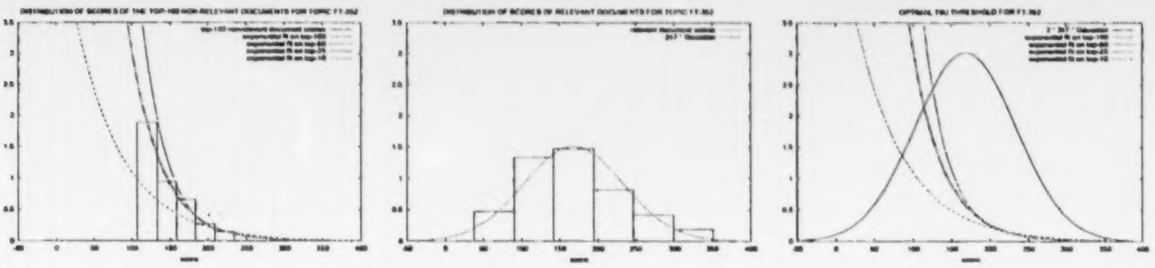


Figure 1: Score distributions and the optimal T9U threshold.

$$= \frac{1}{r} Q_n \, IDF \sum_{i=1}^{r} D_i^T = \frac{1}{r} Q_n \, IDF \, B_n^T \, . \qquad (13)$$

Obviously, the individual document scores $s_1, \ldots, s_r$ are not needed, but only the accumulated sum of the relevant document vectors B_n. Using B_n, the current query Q_n, and the most recent IDF matrix, the mean score can be calculated accurately and incrementally. This way of keeping an average score accurate has been seen before in [6]; we have merely re-formulated it, using matrices, for compactness.

The standard deviation may be obtained from the formula $\sigma_r^2 = \mu_r^{(2)} - \mu_r^2$, where $\mu_r^{(2)}$ is the mean of the squares of the relevant document scores. The proof of the incremental formula for $\mu_r^{(2)}$ is more complex; here we give only the final formula:

$$\mu_r^{(2)} = \frac{1}{r} \left(Q_n \, IDF\right) B_{\text{dyad},n} \left(Q_n \, IDF\right)^T ,$$

$$B_{\text{dyad},n} = \sum_{i=1}^{r} D_i^T D_i \, . \qquad (14)$$

Consequently, a $K \times K$ matrix $B_{\text{dyad},n}$ is required for keeping the deviation of the scores accurate, however, this matrix can be updated incrementally. K grows with a square-root, so $B_{\text{dyad},n}$ grows linearly in time. This makes term selection indispensable (see Section 3.4).

3.5.4 Optimizing T9P

The S-D threshold optimization we have introduced in Section 3.5 can be applied to optimize T9P. However, in this case Eq. 9 does not have analytical solutions, therefore it has to be solved numerically. Regrettably, we did not bother to do that.

The technique we used lowers the threshold after every "quiet" month with respect to how many documents are missing according to the pro-rata adjusted minD value. It goes as follows:

1. right after a query update, start collecting the document scores in the range $[\mu_{\text{nr}}, \theta]$, where μ_{nr} is the mean score of the N^+-documents and θ the optimal S-D threshold for $U = R_+ - N_+$.

2. if after one month of documents nothing is retrieved, calculate how many should have been retrieved by the current time (pro-rata).

3. check how many are missing:
   m = pro-rata−retrieved.

4. if $m > 0$, lower the threshold to s_m, where s_m the top-m score seen below θ.

The method works, in the sense that it retrieves around minD(= 50) documents or more. Moreover, it retrieves the ones that score the highest. It assumes, however, that the distribution of relevant documents in the stream is uniform (or their relative density is approximately constant), in general a false assumption. Another drawback of the method is that it optimizes the threshold for $U = R_+ - N_+$ and not for precision. All of these, we believe, make our submitted T9P runs moderately satisfactory.

After all, we should have at least tried to solve Eq. 9 numerically. Although analytical formulas are mathematically more elegant, in practice, numerical methods are efficient and easy to implement.

3.6 Experiments with FILTERIT

The FILTERIT system presents two features which we are interested in comparing their effectiveness with other systems: the threshold optimization for linear utility functions (see Section 3.5.2), and the decay of training documents (see Section 3.2). The tuning parameters were numerous, and the runs allowed for submission to TREC-9 were limited to 4 for adaptive and to 2 for batch filtering (including batch-adaptive). Moreover, we submitted one of the two batch filtering runs with the LCS system described in Section 4. These limits do not allow extensive comparisons, and some choices had to be made.

Our strategy in deciding what to submit was as follows. For the two of the four adaptive runs we did not use any of the two features but rather conventional techniques. In this way, we expected to have at least two runs with conventional effectiveness, in case our techniques would have failed. The other two adaptive and the single batch filtering run combine all the new features. All parameters were set at "safe" values, as these were determined by our experiments with the Financial Times (FT) collection. More aggressive settings have yielded better effectiveness on FT, however, we do not believe that these generalize in all collections.

3.6.1 Rocchio Parameters, and Initial Query Elimination

All adaptive runs use $\alpha = \beta = \gamma$ for Rocchio. These tasks start with a query and only 2 relevant training documents. In pilot runs on FT, traditional settings with $\alpha < \beta$ seemed to overfit the classifiers on those 2 relevant documents. Therefore, such small training sets should not be trusted and the initial query Q_0 should be weighted fairly high, e.g., as high as $\alpha = \beta$. As a filter is collecting more and more relevant documents, the contribution of the initial query can gradually be eliminated. Consequently, we moreover multiply Q_0 with $10/(R_n + 10)$ while calculating the new query Q_n. We do not use such an *initial query elimination* for the runs with decay since the initial query decays anyway.

For the batch-adaptive task, α is set at the one-fourth of β. Since larger training sets are given for this task, the danger of overfitting is smaller. When using query zones, [12] have shown that $\beta = \gamma$ is a reasonable setting. This also explains why we set $\beta = \gamma$ also for the adaptive tasks. Thresholding document scores during filtering can been seen as a form of *on-the-fly query zoning*. Any non-relevant documents retrieved in this way are indeed the most interfering with the query. This setting has worked out well for us in our experiments on FT.

3.6.2 Submitted Runs

Table 2 summarizes the runs we submitted, their parameter settings, and the final results obtained.

KUNa1T9U and KUNa1T9P do not use decay, term selection, or the threshold optimization described in this article. The threshold per topic is set at the midpoint of the average scores of relevant and non-relevant documents. In fact, for KUNa1T9U we set thresholds at the one-third of the distance between the non-relevant and the relevant mean score to reflect the fact that the gain of retrieving a relevant document is double than the cost of retrieving a non-relevant one (definition of T9U). Therefore, the thresholds should be lower than the midpoints to retrieve more relevant documents.

KUNa2T9U and KUNa2T9P use a decay for training documents with half life set to 2 years; we have found this value reasonable for filtering medical articles. Term selection cutoff is set at the top-500 terms; a light cutoff because our threshold optimization seems to require at least 250 terms in a classifier (Section 3.5.1), and moreoever, long classifiers are necessary when tracking relevance drifts [1]. Thresholds are S-D optimized, however, not exactly as we have described in this article.

Our S-D method was in an early stage at the time of submition. What we did was to approximate the N_+ document scores with a Gaussian. Repeatedly adapting a query causes the distribution of non-relevant retrieved document scores to look more like a bell-shaped distribution. This is an artifact of re-training, however, and does not correspond to what is really happening below the threshold. Nevertheless, it has worked out resonably, suggesting that a Gaussian approximation may be usable since it still gives some estimation of the *spread* of the non-relevant scores; however, it is of dubious accuracy. We will come back to this in Section 3.6.3.

For KUNbaT9U (batch-adaptive) we basically use the same settings as for KUNa2T9U, except for the Rocchio parameters. Moreover, we apply *document sampling* and *query zoning* [12]. The training stream (`ohsumed.87`) consists of around 54,000 documents, and only a few of them are relevant for a topic. For efficiency reasons we do random sampling with probability 0.1 to reduce the number of non-relevant training documents. Then we apply query zoning to select and use for training only the top-r scoring non-relevant documents, where r is the number of relevant training documents. We calculate the query zone with formula 1 for $\gamma = 0$.

The adaptive runs do not show large differences in effectiveness, mainly because of the modest parameter settings for term selection cutoff, half life value, and the fact that the S-D threshold optimization technique is triggered only when at least 5 relevant and 5 non-relevant training documents are made available. Many topics did not reach these numbers, so they were actually filtered with thresholds set at weighted midpoints.

3.6.3 More Runs

In this Section, we provide the extras runs we made in order to find where some the parameters of FILTERIT peak, and determine which techniques actually work. All runs reported here use (unless otherwise is noted): query zoning to select for training only the top-r non-relevant documents, term selection cutoff set at 500, no decay, and thresholds set at weighted midpoints for T9U.

	Runs				
	KUNa1T9U	KUNa2T9U	KUNa1T9P	KUNa2T9P	KUNbaT9U
Task	adaptive				batch-adapt.
Rocchio	$\alpha = \beta = \gamma$				$4\alpha = \beta = \gamma$
Q zoning	—				top-r
Q_0 elimination	$10/(10 + R_n)$	no	$10/(10 + R_n)$	no	no
T-opt. for	T9U	T9U	T9P	T9P	T9U
T-opt. method	$(\mu_r + 2\mu_{nr})/3$	S-D	$(\mu_r + \mu_{nr})/2$	S-D	S-D
half life	∞	2 yrs	∞	2 yrs	2 yrs
T.S. cutoff	—	500	—	500	500
Result	+16.8	+17.3	0.258	0.231	+19.4

Table 2: Parameter settings for adaptive and batch-adaptive submitted runs.

Document Sampling and Query Zoning. We have investigated the effect of sampling the non-relevant document space. We have run a batch-adaptive task with 3 different samples. Table 3 presents T9U and F-measure results. All samples are made by selecting randomly one out of ten non-relevant training documents from `ohsumed.87`. Then query zoning is applied before training the initial classifier. The results do not show significant differences.

sample	T9U	F_1
A (official)	19.5	0.406
B	19.8	0.406
C	19.1	0.403

Table 3: The effect of sampling the non-relevant training document space.

Term Selection. Figure 2 shows the impact of our term selection method (see Section 3.4) for different cutoff values. The runs are batch-adaptive using sample A. The average T9U seems to peak between 500 and 125 terms.

Decay. We have experimented with different half-life values on an adaptive task. Figure 3 shows that the average T9U peaks somewhere between 2 and 8 years of half life. However, further analysis has revealed that effectiveness peaks at considerably different half-life values across topics. An optimization of half-life per topic — if we only had a way to do that — would have resulted in great improvements of the average T9U.

Threshold Optimization. In [13] we give the TREC-9 evaluation table of our submitted batch and batch-adaptive runs. We have made a supplemental batch-adaptive run with the revised S-D threshold optimization as described in this paper, i.e. by fitting an exponential on the top-50 non-relevant training documents[7]. When the non-relevant training document buffer exceeds 50 documents, we sort them according to their scores and discard the lowest scoring one. The results are presented in the last column, labeled as `FilterIt-ba`. They show an improvement in the average T9U from 19.4 to 21.3.

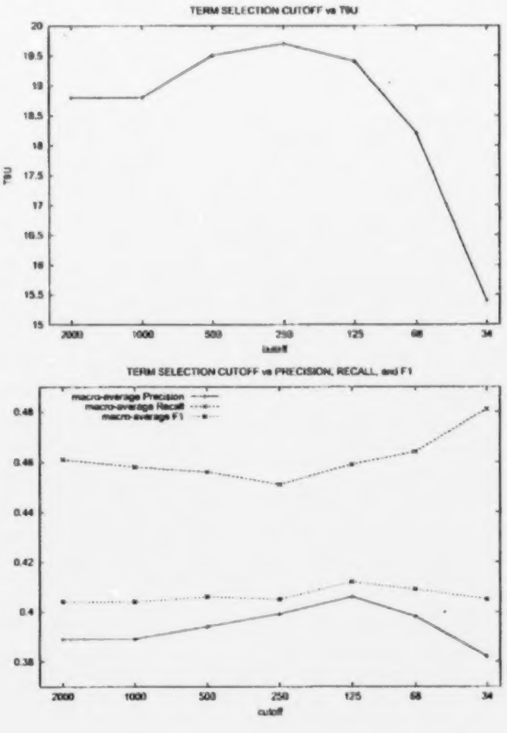


Figure 2: The effect of term selection.

---

[7] Note that we have not optimized any other parameter according to our post-official runs; we have merely used a better S-D optimization.

BEST COPY AVAILABLE

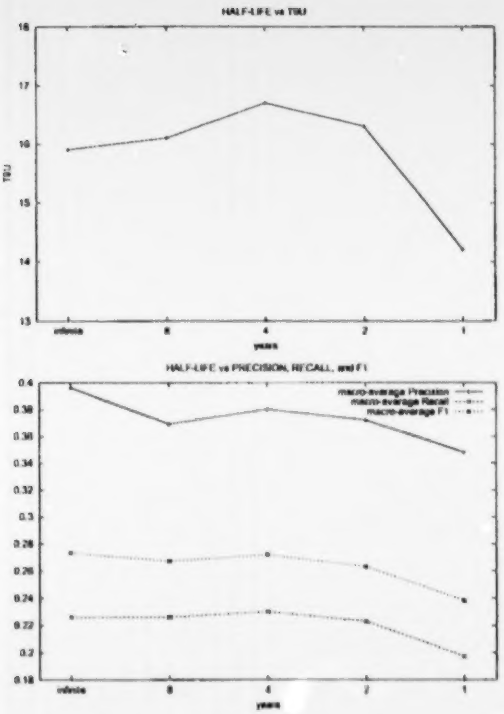


Figure 3: The effect of decay.

One could argue that setting thresholds with the weighted midpoint method works out comparably to the S-D optimization (compare e.g. KUNa1T9U to KUNa2T9U), but this is not the case. In fact, the good performance of the weighted midpoint method has been purely accidental; the same goes for the aforementioned Gaussian fit on non-relevant document scores. The mean score of non-relevant documents μ_{nr} has been estimated on the top-scoring non-relevant documents. This produces a relatively large μ_{nr}, which in its turn results in tight thresholding. When we have tried to increase the number of non-relevant documents, the weighted midpoint method as well as the Gaussian fit have greatly failed: the more non-relevant documents are used for training, the lower the μ_{nr}, thus lower thresholds. The methods fall too easily into the *selectivity trap* of retrieving too many (mostly non-relevant) documents. The revised S-D optimization as described in this article has proved much more reliable and robust in a range of settings, consistently avoiding such selectivity traps.

4 The LCS System

The routing and batch filtering tasks were carried out by the LCS system[8] [9]. The system is based on the Winnow mistake-driven learning algorithm [8]. The Winnow algorithm has, to our knowledge, not been used before in TREC. It can cope well with large numbers of terms, which is certainly the case here: after pre-processing, the training set had some 52,000 different terms.

4.1 The Winnow Algorithm and Improvements

During training, the *Balanced Winnow* algorithm [8, 7] iteratively computes two weights $w^+_{i,C}$ and $w^-_{i,C}$ for every term i and class (topic) C. These *winnow weights* are used to compute the score $S(D,C)$ of a document D for the class C as:

$$S(D,C) = \sum_{i \in D} (w^+_{i,C} - w^-_{i,C}) * u_{i,D} \,, \qquad (15)$$

where $u_{i,D}$ is the *term strength* (weight) of term i in document D. Classification is achieved by thresholding $S(D,C)$ using a threshold θ.

Winnow is *mistake-driven* in the sense that it adjusts the weights $w^+_{i,C}$ and $w^-_{i,C}$ only if their current value, during an iteration, leads to a misclassification. If a relevant document scores below θ, then the winnow weights for the terms occurring in the document are multiplicatively updated using a promotion factor Alpha. Similarly, for a non-relevant document scoring above θ, the weights are demoted using a demotion factor Beta. The threshold θ is considered fixed, and the learning stops when there are no weight updates during an iteration, or earlier even in order to avoid over-training. Topic descriptions were considered as normal documents, since Winnow provides no special mechanism for dealing with requests.

The implementation of Winnow in LCS is similar to the one described in [7], with two modifications:

1. the document terms $u_{i,D}$ are *ltc* weighted [5], without the vector length normalization factor. Traditionally, $u_{i,D}$ are set either to the frequency of i within D, or to the square-root of the frequency. In experiments on the FT corpus, *ltc* has proved to work definitely better than the former, and slightly better than the latter.

2. Winnow weights were initialized for training as:

$$w^+_{i,C} = \frac{2\theta}{ADS} \,, \quad w^-_{i,C} = \frac{\theta}{ADS} \,,$$

---

[8] Esprit project DOcument ROuting (DORO), `http://www.cs.kun.nl/doro`

BEST COPY AVAILABLE

$$ADS = AVG_D \frac{\sum_{i \in D} u_{i,D}}{size(D)},$$

where $size(D)$ is the number of unique terms in document D. This initialization improves Winnow's convergence speed.

The convergence speed of the Winnow algorithm (the number of iterations needed to learn a stable classifier) depends rather critically on the initial values of the weights. In [7], all positive weights are initialized as θ/d, where θ is the threshold and d the average number of "active features" in documents. This choice ignores collection statistics for terms. In our initialization, an average document obtains an initial score equal to θ. Since term strengths are taken into account, fewer iterations are needed.

4.2 Threshold Setting by Cross-evaluation

The Winnow algorithm has a "natural" threshold $\theta = 1.0$ for separating relevant from non-relevant documents, putting equal importance on precision and recall. T9U stresses recall more than precision, however. The S-D threshold optimization, as described in section 3.5, has not (yet) been implemented in the LCS, so the necessary threshold optimization was performed by *cross-evaluation*.

The training set (`ohsumed.87`) was split into n subsets of the same size, which each in turn was used as *optimization test set* while all the other subsets, together with the topic descriptions, were used as *optimization training set*. The scrap of the split was included into the optimization test set. After training Winnow with $n-1$ subsets, the documents of the remaining subset (optimization test set) were ranked according to their scores. Then, by going down the rank, the threshold value that optimized T9U was found. We performed the cross-evaluation for $n = 2$, 3 and 4, and we took the mean of all $(2 + 3 + 4 = 9)$ optimal threshold values.

4.3 Experiments with LCS

4.3.1 Submitted Runs

We set the Winnow parameters to the values that gave the best results on the FT corpus (Table 4). We use the *thick separator* heuristic [7]: instead of a single threshold θ, a *threshold range* $[\theta^- : \theta^+]$ is used. There is a promotion whenever a relevant document obtains a score below θ^+ and a demotion when a non relevant document gets a score over θ^-. This heuristic achieves a better separation between relevant and non-relevant documents. The asymmetry around the standard threshold (1.0) forces the algorithm to perform more promotions than demotions on the early iterations. This compensates for the asymmetry between the numbers of relevant and non-relevant training documents, speeding up convergence.

parameter	value
Alpha	1.1
Beta	0.9
ThresholdRange	on
θ^+	1.3
θ^-	0.9
MaxIters	30

Table 4: Winnow Parameters.

We have submitted 2 routing runs, KUNr1 and KUNr2. LCS has originally been developed for *mono-classification* tasks, i.e., each document belongs to exactly one class. This means that the relevant training documents for one class are considered as non-relevant training documents for all other classes. That is certainly not the case in filtering, so we had to do separate runs per topic assuming two classes: relevant and non-relevant. The routing results KUNr1 were produced like this.

The approach of separate runs is correct but obviously inefficient. So, we also tried to process all topics at once (KUNr2), hoping that they do not have relevant documents in common, or even if they do, the impact of this dubious approach on effectiveness would not be that great. Luckily, in the given dataset, it was not: the average uninterpolated precision was practically the same. We obtained 0.237 for KUNr1 and 0.234 for KUNr2.

The batch filtering run KUNb was obtained through the thresholding of the rankings of KUNr1. Thresholding was performed by the cross-evaluation method we described in Section 4.2.

4.3.2 More Runs

The KUNb results, obtained with separate thresholds per topic calculated by cross-evaluation, can be compared with those obtained by a simpler method: a uniform threshold for all topics. We can choose as a uniform threshold any value in the threshold range; such a choice should give the same result if the classification is perfect. But two values are special: 1.0 (average document score before training), and 1.1 (the center of the threshold range).

Table 5 shows that the results for $\theta = 1.0$ are worse than those for $\theta = 1.1$. Moreover, a uniform threshold set at 1.1 gives slightly better results than the separate thresholds computed by cross-evaluation. It seems that the cross-evaluation method has failed, mainly because the training sets had relatively small numbers of relevant training documents. Splitting the sets for cross-evaluation, made the things even worse.

Run	T9U
separate θ's via cross-evaluation (KUNb)	5.0
uniform $\theta = 1.0$	−3.5
uniform $\theta = 1.1$	6.0
best possible thresholdings on KUNr1	17.9

Table 5: Different thresholdings on Winnow.

The best possible thresholdings on the rankings of KUNr1 would have obtained an average T9U of 17.9; not very great either, considering that the largest possible average T9U for the given test set is 104.9. This implies that the rankings achieved are not very good.

5 Overall Comparison and Discussion

In [14] we give the TREC-9 evaluation table of our submitted routing runs with the LCS system. The rightmost column, `FilterIt-r`, corresponds to a supplemental routing run with the FilterIt system. Obviously, FilterIt gives better rankings than LCS; the corresponding average uninterpolated precision figures are 0.373 and 0.237. Thresholding the rankings of `FilterIt-r` with the optimal S-D thresholds (as these were estimated by the method described in section 3.5.2) we obtained a (non-adaptive) batch run with FilterIt. Its results are presented under the label `FilterIt-b` in [13]. An average T9U of 14.8 is obtained in contrast to 5.0 obtained by LCS.

Since `FilterIt-r` and `FilterIt-b` are not *post-factum* optimized, it seems that we should have submitted all runs, for all filtering tasks, with the FilterIt system. The `FilterIt-r` routing run, with an average precision of 0.373, would have ranked us as second best system; the first system scored at 0.385. The `FilterIt-b` batch run, with an average T9U of 14.8, would have ranked us clearly as the best system; the best official batch run scored at 7.5. The official TREC-9 comparison tables of the tasks we have participated can be found in [15]. At an rate, we are very satisfied with the performance of the FilterIt system in our official runs. We have clearly achieved the best scores in all adaptive and batch-adaptive tasks optimized for T9U. Compare the 17.3 (KUNa2T9U) and 19.4 (KUNbaT9U) to the 10.7 and 13.6 of the second best systems in the corresponding tasks. The official T9P runs are also satisfactory; our best run has achieved 0.258 (KUNa1T9P), a rather comparable effectiveness to the 0.294 of the best system. After all, we have not optimized exactly for T9P, but for some other related utility measure, in order to simplify the calculations (see Section 3.5.4).

Why are the results with the LCS less satisfactory? According to our experience, Winnow performs better than Rocchio when large numbers (hundreds) of relevant training documents are available for each class. This was not the case in the batch and routing tasks of TREC-9 where some topics had very few relevant training documents. This may largely be responsible for Winnow's weak performance. Furthermore, with 30 iterations in the learning phase, there is some evidence of overtraining.

Why are the results with FilterIt so good? Let us summarize the methods we have used: accurate and incremental adaptivity as soon as a single training document becomes available (in contrast to re-training in batches), local adaptivity (training documents of decaying value in time), on-the-fly term selection (in contrast to just cutting off classifiers), the S-D threshold optimization (note that we are talking about "optimization" rather than "setting"), and initial query elimination. Moreover, all parameter settings (e.g. Rocchio's α, β, γ, term selection cutoff, half life) have either been empirically determined on the Financial Times collection or at least motivated. There is evidence as well that *Ltu* weighting and query zoning have contributed considerably to effectiveness. The FilterIt system is a typical example of: *the whole is more than the sum of its parts.*

6 Conclusions and Further Research

In this first-time contribution to TREC, we have focussed mainly on the adaptive tasks. Our contribution to adaptive filtering has been threefold:

- We have investigated the value of retrieved documents as training examples in relation to their time of retrieval. For this purpose, we have introduced the notion of the *half-life* of a training document. The approach has presented promising results.
- We have introduced the *Score-Distributional (S-D) threshold optimization* method, capable of optimizing any effectiveness measure defined in terms of the traditional contingency table. The method has found to be very effective, and it can moreover be applied incrementally.
- We have developed a system that allows *incremental adaptivity*, minimizing its computational and memory requirements without sacrificing too much of accuracy.

In overall, we are very satisfied with our adaptive results; we have clearly achieved the best utility scores in all adaptive and batch-adaptive tasks that we have participated.

The results of the batch and routing tasks are less satisfactory, but at least the feasibility of using the Winnow algorithm in these applications has been demonstrated.

Summarizing, our TREC-9 participation has motivated a great deal of research. As a result, we have finalized the S-D threshold optimization in [4], and we have reconsidered the nature of the filtering task in [1]. Our plans for further research include: finding a way of detecting relevance drifts in order to select appropriate half life values, and to revise the term selection method we have introduced in [3].

Acknowledgments

The first (and main) author would like to thank André van Hameren for the fruitful discussions during the development of the FILTERIT system.

References

[1] A. Arampatzis and T. P. van der Weide. Document Filtering as an Adaptive and Temporally-dependent Process. Technical Report CSI-R0103, University of Nijmegen, January 2001.
Available from `http://www.cs.kun.nl/~avgerino`.

[2] A. Arampatzis, T. P. van der Weide, C. H. A. Koster, and P. van Bommel. Linguistically Motivated Information Retrieval. In A. Kent, editor, *Encyclopedia of Library and Information Science*. Marcel Dekker, Inc., New York, Basel, 2000. To appear.
Available from `http://www.cs.kun.nl/~avgerino`.

[3] A. Arampatzis, T. P. van der Weide, C. H. A. Koster, and P. van Bommel. Term Selection for Filtering based on Distribution of Terms over Time. In *Proceedings of RIAO'2000 Content-Based Multimedia Information Access*, pages 1221–1237, Collège de France, Paris, France, April 12–14 2000.
Available from `http://www.cs.kun.nl/~avgerino`.

[4] A. Arampatzis and A. van Hameren. The Score-Distributional Threshold Optimization for Adaptive Binary Classification Tasks. Technical Report CSI-R0105, University of Nijmegen, January 2001.
Available from `http://www.cs.kun.nl/~avgerino`.

[5] C. Buckley, G. Salton, and J. Allan. The Effect of Adding Relevance Information in a Relevance Feedback Environment. In W. B. Croft and C. J. van Rijsbergen, editors, *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 292–300, Dublin, Ireland, June 1994. ACM Press.

[6] J. Callan. Learning While Filtering Documents. In W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson, and J. Zobel, editors, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 224–231, Melbourne, Australia, August 1998. ACM Press, New York.

[7] I. Dagan, Y. Karov, and D. Roth. Mistake-driven Learning in Text Categorization. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, 1997.

[8] N. Littlestone. Learning Quickly when Irrelevant Attributes Abound: a New Linear-threshold Algorithm. *Machine Learning*, 2:285–318, 1988.

[9] H. Ragas and C. H. A. Koster. Four Text Classification Algorithms Compared on a Dutch Corpus. In W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson, and J. Zobel, editors, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 369–370, Melbourne, Australia, August 1998. ACM Press, New York.

[10] J. J. Rocchio. Relevance Feedback in Information Retrieval. In *The SMART Retrieval System — Experiments in Automatic Document Processing*, pages 313–323, Englewood Cliffs, NJ, 1971. Prentice Hall, Inc.

[11] A. Singhal. AT&T at TREC-6. In E. M. Voorhees and D. K. Harman, editors, *The Sixth Text REtrieval Conference (TREC-6)*, pages 215–225, Gaithersburg, Maryland, November 19–21 1997. Department of Commerce, National Institute of Standards and Technology (NIST) Special Publication 500-240.

[12] A. Singhal, C. Buckley, and M. Mitra. Learning Routing Queries in a Query Zone. In N. Belkin, D. Narasimhalu, and P. Willett, editors, *Proceedings of the 20st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 25–32. ACM Press, New York, July 1997.

[13] `http://www.cs.kun.nl/~avgerino/Avi_Arampatzis/publications/TREC9/batchT9U.txt` .

[14] `http://www.cs.kun.nl/~avgerino/Avi_Arampatzis/publications/TREC9/routing.txt` .

[15] `http://www.cs.kun.nl/~avgerino/Avi_Arampatzis/publications/TREC9/averages.txt` .

Information Space based on HTML Structure

Gregory B. Newby[*]
UNC Chapel Hill

Abstract

The main goal for the Information Space system for TREC9 was early precision. To facilitate this, an emphasis was placed on seeking matches from only the *TITLE, H1, H2 and H3 tags in the Web (wt10G)* and large Web (wt100) document collections. Ranking of documents was based on a combination of Boolean union sets, term weights, and principal components analysis (PCA). Very large sparse cooccurrence matrices were created for term weighting and PCA. The Information Space system is part of a larger general software package called IRTools.

Introduction

This year's TREC entry for the Information Space system builds on past years, with some specific goals. Due to 2000 being the first year with Web data for the main task for TREC (instead of newswire and other data, as in past years), it seemed desirable to make use of the structure of HTML. As casual observation of the popular Web search engines (Google, Lycos, etc.) reveals, these systems provide additional weight to terms occurring in the <TITLE> tags of documents, in addition to searching through the terms in each document.

The Information Space (IS) main Web task entry for this year focused only on tags in the <TITLE>, <H1>, <H2>, and <H3> tags in the datasets. This was intended to facilitate early precision, by matching the short TREC topic title or title plus description statements to terms in these tags. The submission for the main Web task was 6 days late, and therefore not judged (although it was counted by NIST as an "official" run). Post hoc analysis of some queries indicate that if results were judged, they probably would not have been substantially better than the non-judged results found in the conference proceedings.

IS also made an entry to the large Web task or VLC. The 100GB VLC (w100) was processed similarly to the main Web task, by focusing only on terms in the same set of tags (title, h1, h2 and h3). Because this run was also submitted late, by nearly 2 weeks, it was not judged. Due to the small number of official VLC submissions and small number of judged documents, no useful recall or precision statistics are available.

This paper will present an overview of the procedure used to index and retrieve from the wt10g and w100 datasets, followed by a brief discussion of the large co-occurrence matrices generated. Then, system-based and relevance-based performance outcomes are discussed. It is concluded that query expansion did not serve well to facilitate early high precision. Furthermore, a lack of sophisticated term weighting also hurt results.

---

[*] Contact data: gbnewby@ils.unc.edu, http://ils.unc.edu/gbnewby. CB 3360 Manning Hall, Chapel Hill, NC, 27599-3360 USA.

The IRTools Software

IS is part of a set of software tools for IR experimentation under development by the author and his colleagues. The software is called the "Information Retrieval Toolkit," or IRTools. The purpose of IRTools is twofold:

1. To provide an integrated collection of C++ classes designed to facilitate IR experimentation; and
2. To incorporate design for large-scale practical use.

Although modern information scientists have always relied on software for their experiments, relatively few have chosen to make their software freely available to others. For those who have shared, the software is often not suitable for re-use in other experimental settings – due to either lack of documentation, cross-platform instability, or non-modular design. IRTools is intended to help address the shortage of software for retrieval experimentation.

Another problem that has often hindered information scientists is the difficulty of demonstrating the scalability of their ideas. IRTools places and emphasis on high performance data structures, file structures and algorithms (Newby, 2000b). Real-world functionality will include the ability to update the document collection (e.g., by spidering the Web periodically). IRTools' goal is to index billions of documents, with hundreds of millions of unique terms, and over a terabyte of aggregated data.

IRTools is designed modularly, as a library of C++ classes. Currently, IRTools is over 25,000 lines of code including test programs. It makes extensive use of the standard template library (STL). The plan for IRTools is to incorporate the functionality of all major types of experimental IR: probabilistic retrieval, the vector space model, latent semantic indexing, simple Boolean retrieval, and others. IRTools will make it easier and faster for information scientists to perform experiments or expand software. The software development is supported in part by a grant from the NSF under their information technology and research (ITR) program. The project homepage is http://irtools.sourceforge.net.

Information Space Techniques for TREC9

Information Space, or IS, is an approach to information retrieval that is similar to latent semantic indexing (LSI). Over the past several years, IS has incorporated different specific techniques to achieve particular goals. IRTools will enable more of these goals to be integrated – for example, the TREC9 IS programs did not have good facilities for term weighting, even though the utility of term weighting using IS techniques was demonstrated in TREC8 (Newby, 2000a; Newby 1998).

The main distinction between LSI and IS is that LSI utilizes a singular value decomposition (SVD) on the term by document matrix, while IS utilizes principal components analysis (PCA) on the term by term matrix. In both LSI and IS, the distinguishing point from the vector space model (VSM) is that terms are not assumed to be mutually unrelated. The basic process is the same, however: document vectors are computed based on the vectors for terms they contain. A query vector is similarly computed, and the closest documents to the query are retrieved.

Although LSI and IS are comparable, and have a similar intellectual heritage in the mathematics of linear algebra, they actually operationalize a significantly different goal. With both LSI and IS, only *k* columns of the eigenvectors from the SVD or PCA process are used, rather than all *N* columns for each of the *N* terms. With LSI, all columns of the eigenvectors would in fact result in a vector space in which all terms are mutually orthogonal – in other words, the same fundamental model of the VSM. Thus, the *k*-dimensional vector space representing term relations in LSI is an approximation of an orthogonal term space. By reducing *k*, LSI attempts to account for assumed "errors" in the original term by document matrix.

With IS, all *N* columns of the eigenvectors would result in a vector space in which term relations are identically scaled to the numeric relations among terms in the original term by term input co-occurrence matrix. Thus, the *k*-dimensional vector space representing term relations in IS is an approximation of the relations among terms actually measured in the term by term matrix.

These differences are moderated by the other differences in how the techniques are actually applied. For most purposes, it is accurate to characterize IS as similar to LSI. The author has written a more extensive treatment of this subject which has been submitted elsewhere for publication.

The specific techniques used for both the main Web and VLC in TREC9 are as follows:

Phase 1: Indexing

1. Only terms in the <TITLE>, <H1>, <H2> and <H3> tags were processed. All terms in other tags were ignored, as was any document metadata for the wt10g or w100 collections. Documents without these tags were ignored.
2. All terms with fewer than 20 characters and consisting only of alphabetical characters A-Z (case insensitive) were indexed. No stemming was applied.
3. A term by term co-occurrence matrix was built for all the indexed terms for all the documents they occurred in. This resulted in a very large and very sparse matrix.

Phase 2: Retrieval

1. Only terms that had been indexed were used; others were stopped. In addition, the SMART stoplist was employed, along with a few additional stop words consisting of HTML tags.
2. Query terms were expanded (by 100 terms for wt10g, and 25 terms for w100). The top co-occurring terms for each query term were added to the query.

3. All documents with any of the expanded query terms were selected for further consideration; the rest of the documents were assumed to be non-relevant.
4. The full (sparse) co-occurrence matrix for all of the expanded query terms was used to calculate the full (dense) correlation matrix for the terms.
5. PCA was performed on this correlation matrix:
    a. The eigenvectors of the correlation matrix were computed
    b. Term vectors were computed as the dot product of that term's eigenvector and the terms standardized ($\acute{z}$) scores from the original co-occurrence matrix.
6. Each document under consideration was located at the geometric center of the expanded query terms it contained (terms it contained that were not part of the expanded query were ignored).
7. The query was located at the geometric center of its terms.
8. The query and document locations were normalized to unit length.
9. Distances from each document to the query were ranked, and the closest retrieved.

Note that the choice of the geometric distance versus cosine is arbitrary for unit length vectors: the ranking is the same. But for non-uniform vector lengths, the geometric distance is more accurate than the cosine, as the cosine only considers the angle of incidence between vectors, not the difference.

Large Co-Occurrence Matrices

A difficulty of working with co-occurrence matrices with large numbers of terms is that the number of updates to the matrix during indexing can be daunting. Consider that for a document with 1000 terms, (1000 x 1000-1)/2 or 499500 term pairs exist, and must be considered for updating the term by term co-occurrence matrix. Even if term ordering or term counts are ignored, the number of possible term pairs per document can be large.

One approach to avoiding a very large number of term pairs for each document is to consider co-occurrence only within subdocuments (this is also conceptually appealing). A subdocument might be considered as a term plus its surrounding terms (a sliding window), terms within the same paragraph, or terms within the same sentence. Another obvious approach, employed by IS for TREC9, is to only consider terms within the same tag set. Here, the co-occurrence matrix was computed based only on terms that were found together within a title, h1, h2 or h3 tag.

This resulted in a manageable number of term pairs for most documents, as HTML titles and h1, h2 and h3 tags tend to contain fewer than a dozen terms. This also added to the sparsity of the matrix, which helps with storage. Were every cell in a term by term matrix to be filled, the storage size on disk would be N times N (for N terms) times the size of each datum stored. For the 1.2M unique terms identified in the w100 collection and 4 bytes per integer, this is well over 5 terabytes.

Using a variation on the Harwell-Boeing sparse matrix format, IS only stored the non-zero cells on disk. The storage required using the IS variation on the H-B format is:

$$S(3*N + 2(C) + 2(N))$$

where:

S is the number of bytes per integer
N is the number of terms (aka rows)
C is the total number of non-zero column entries

Using this format with the number of non-zero co-occurrence scores reported in Table 1, about 304Mbytes were required to store values for the co-occurrence matrix for the 1.2M unique terms from w100, a savings of well over 99%. In fact, this is nearly twice as much storage would be required to store only ½ of the matrix with no loss of information, as the matrix is symmetric. Both sides were used during the retrieval phase described above, so the symmetric matrix was converted to a full matrix after indexing was completed.

Table 1: Term co-occurrence matrix properties

Dataset	Term count	Non-zero co-occurrence scores	Sparsity
wt10g	310050	27233214	0.00028329
w100	1207560	34982212	0.00002399

Indexing and Retrieval System-Based Performance Measures

For TREC8, IS was able to index w100 in 5 hours, and process all 10K VLC queries in about 52 seconds. The TREC9 implementation did not strive for such high system performance measures: term co-occurrence added significantly to the indexing overhead, as did identification of tag sets within documents. Indexing time for the w100 was about 120 hours; the wt10g took about 20 hours.

As for TREC8, all indexing and retrieval was completed on UNC's Sun Enterprise Server 10000, a high-end server that was shared with many other processes. The ES10000 had 36 processors and 20GB of memory, but IS utilized only one processor at a time and operated in less than 2GB of memory. A high-speed disk subsystem with a tape-to-disk robot enabled virtually unlimited storage with latency of less than a minute for staging the files to be indexed.

Retrieval for the wt10g took well under .1 seconds per query. Query processing involved minimal disk access: the key to the inverted index was read into memory, as was the term hash and full co-occurrence matrix. Disk access was needed to get inverted index entries (that is, the list of documents containing each expanded term) and to map document ID numbers to TREC document strings.

For the w100, retrieval time depended on what sort of query expansion was used. When simple query expansion by 25 terms was used, as described above, queries were completed in an average of .21 seconds across the 10K topic statements. A more sophisticated query expansion model was attempted, in which several iterations and permutations on the co-occurrence matrix were made. The retrieval performance for this variation is not known, because the w100 runs were not judged, but the system performance of over 9 seconds per query is not favorable.

Table 2: System performance for indexing and retrieval

Build index	wt10g	20 hours	
	w100	120 hours	
Index size	wt10g	.58GB	
	w100	2.7GB	
Retrieval time	wt10g	.1 sec/query	
	w100	.21 sec/query	method 1: simple expansion
	w100	9.7 sec/query	method 2: complicated expansion

Retrieval Performance

Because the results for wt10g were not judged, there is some risk of bias in interpretation of the TREC performance measures. However, an informal evaluation of non-judged documents for a set of 6 topics gave the author some confidence that the retrieval performance measures are reasonably indicative of IS' performance in TREC9.

Because there are essentially no judgments for the VLC that are useful for evaluating the w100 submission discussed above, no retrieval performance measures can be discussed here.

For the main Web task, recall from above that the main goal for this year's work was to have high early precision by utilizing the structure of HTML documents. The reasoning was that terms in the title, h1, h2 and h3 tags were most indicative of a document's content. Thus, indexing and retrieval focused on terms in those tags.

In hindsight, it was poor judgment to apply query expansion. In reading through highly-ranked documents, many documents had expanded terms but no query terms. More effective term weighting would have helped avoid this problem, although computation of term weights was hindered by the particular file structures employed (because counts of the frequency of term occurrences were not kept at a document level, only a tag level).

A better approach would have been to bypass the use of the co-occurrence matrix entirely in order to develop baseline retrieval performance. In other words, to perform simple ranked Boolean retrieval based only on terms occurring in the targeted tag sets. Although this would have resulted in several TREC9 topics with no results, a far larger dataset (either w100 or, more interestingly, the Web as a whole), presumably would have produced results for all 50 topic statements.

A challenge in seeking strong retrieval performance combined with strong system-based performance measures is the conflict in the number of documents that can be evaluated. Conceptually, IS (like LSI) would like to evaluate the relationship between every single document in the collection and a query. This is because the IS technique (like LSI) enables matching based on concepts even when terms do not match. However, for practical purposes this is not feasible: evaluating all 18M w100 documents would take too long.

There may be a solution to managing the size of the problem for computing all possible document relations, as discussed in the author's submission to the TREC8 proceedings. But in the meantime, the time-tested approach for IR is to only consider the subset of documents that contain terms of interest – either the query terms themselves, or the query terms plus expanded terms.

Based on the previous paragraphs, the IS system was implemented to evaluated a larger subset of documents than would be evaluated based on a simple Boolean matching of query terms, but far smaller than the complete document set. This is a goal consistent with traditional goals of the IS approach, but (again, in hindsight) probably not a good match for efforts at high early precision based on a limited number of HTML tags.

The specifics of retrieval performance are as follows. For wt10g, four variations on the steps described above were submitted:

1. iswt: title-only
2. iswtd: title + description
3. iswtdn: title + description + narrative
4. isnnwt: title + description + narrative, but with "not" or "non-relevant" phrases automatically removed

Retrieval performance for all four sets was not outstanding. Table 3 shows that the overall number of relevant documents retrieved @ 1000 is fairly low, with under 10% of relevant documents identified by any set. Intuitively, this would be the retrieval performance statistic most likely to be hurt by non-judged sets.

Table 3: Relevant retrieved @ 1000

	iswt	iswtd	iswtdn	isnnwt
Best	1	2	2	1
>= Median	3	3	4	1
Worst	12	11	13	24
Total relevant retrieved	242	236	172	126
% total relevant retrieved	9.25%	9.02%	6.57%	4.81%

Retrieval performance based on average precision tells approximately the same tale. IS tended to have scores above the median when the median scores were relatively low, without ever achieving average precision over 0.33.

Table 4: Average precision

	iswt	iswtd	iswtdn	isnnwt
Best	1	2	2	1
>= Median	8	7	7	5
Worst	13	12	12	24

What of early precision? Precision at 10 docs (P@10) across the 4 sets was not as high as hoped. None of the sets achieved perfect precision at 5 or 10 documents. Fewer than ½ of the queries for all sets resulted in any relevant documents at all in the top 10, which is disappointing. However, as shown in Table 5, these were numerous queries with numbers of relevant documents in the top 5 or 10 documents presented.

Table 5: Precision at 5 and 10 documents

P@5 score	iswt	iswtd	iswtdn	Isnnwt
0.8	0	1	0	0
0.6	1	2	3	0
0.4	4	7	3	6
0.2	12	13	16	11

P@10 score	iswt	iswtd	iswtdn	isnnwt
0.5	0	1	0	0
0.4	2	1	2	0
0.3	4	10	4	2
0.2	5	1	4	4
0.1	11	16	14	13

The main trends evident from examining the TREC9 topics and IS retrieval performance are variability in the HTML document use of tags, and failure of query expansion. Variability is, as mentioned above, perhaps less of a problem in a larger dataset (w100 or the whole Web). Exact matches of title or title + description terms were fairly rare. Furthermore, more effective retrieval would necessitate additional examination of the terms within the documents, not only the four tags used here.

From this result, we tentatively conclude that better retrieval from HTML documents would involve multiple phases or ranking schemes. At one level, documents with matching <TITLE> or other key HTML tags should be given high consideration. At another level, more typical IR techniques should be employed in order to identify potentially useful documents that do not have the query terms in the <TITLE> or other targeted tags. Then, ranking schemes need to be developed to assess which documents from these two sets of candidates are best for retrieval.

For query expansion, as mentioned above, the danger is in retrieving documents on unrelated topics due to the variability in human language. There is little reason to doubt the general utility of query expansion based on the results here, and in fact prior IS entries to TREC have discussed the utility of the term correlation matrix for identifying synonyms.

For query expansion, we suggest that relatively inexpensive approaches, such as the co-occurrence matrix applied here, must be used with caution. More expensive approaches would, presumably, result in fewer ambiguous terms being used – such approaches might be applied at the indexing phase, the query phase, or the document ranking phase. Approaches could include dictionary lookups of term meanings and relations, more detailed statistical analysis (including LSI), and part of speech tagging. In fact, all three approaches and other variations have been used by IS in the past, and will be incorporated for further experimentation in IRTools.

Conclusion

Early precision was not achieved to the extent hoped for. The main problems were query expansion, which added some inappropriate terms to some topic statements, and reliance on only the <TITLE>, <H1>, <H2> and <H3> tags. For future work, terms from other tags will be included in the index, and query expansion will be employed more selectively.

Continued development of IRTools and the IS techniques it contains is anticipated to make it easier to incorporate multiple techniques without a large investment in programming time. A comparison of the relative contributions of the effects of such factors as stemming, PCA and LSI techniques, query expansion, term weighting and other approaches is needed to assess the situations in which each technique is most important for high precision or other goals.

References

Newby, Gregory B. 2000a. "Moving More Quickly Towards Full Term Relations in Information Space." Text REtrieval Conference (TREC-8) Proceedings. Gaithersburg, MD: National Institute of Science and Technology. November 16-19, 1999.

Newby, Gregory B. 2000b. "The Science of Large-Scale Information Retrieval." Internet Archive 2000 Colloquium. San Francisco, March 8-9.

Newby, Gregory B. 1999. "Information Space Gets Normal." Text REtrieval Conference (TREC-7) Proceedings, pp. 567-571. Gaithersburg, MD: National Institute of Science and Technology. November 9-11, 1998.

Web Document Retrieval using Passage Retrieval, Connectivity Information, and Automatic Link Weighting – TREC-9 Report

Franco Crivellari Massimo Melucci[*]

Department of Electronics and Computer Science
University of Padova (Italy)

Abstract

This report describes the participation at the Web track of the TREC-9 of the Information Management Systems research group of the Department of Electronics and Computer Science at the University of Padova (Italy). TREC-9 has been our first participation to TREC and, then, to the Web track. In the following, we describe the experimental approach we have chosen, the research hypotheses and questions, the problems we encountered, the results we reached and our conclusions. We consider this experience as the first step towards the participation to the next Web tracks.

1 Experimental Approach

The approach we have taken to address the problems and the research questions regards both the scientific side and the implementation side. As regards to the scientific side, we employed an experimental approach that mixes both classical advanced information retrieval (IR) techniques, and connectivity-based algorithms for IR on the Web. Figure 1 depicts the whole process being described below. Specifically, we have chosen those classical IR techniques, i.e. passage retrieval and blind relevance feedback, which have proven to be effective to produce good retrieval results [1]. Moreover, we are interested to test whether the connectivity-based algorithms, which have been proposed in different Web contexts, are effective tools to improve classical techniques. As regards to the implementation side, we developed in-house software and employed other software modules that are publicly available.

---

[*] Correspondence author: Dipartimento di Elettronica e Informatica – Via Gradenigo, 6/A – 35131 Padova – Italy – E-mail: melo@dei.unipd.it – Telephone: +39-049-827-7802 – Fax: +39-049-827-7826

BEST COPY AVAILABLE

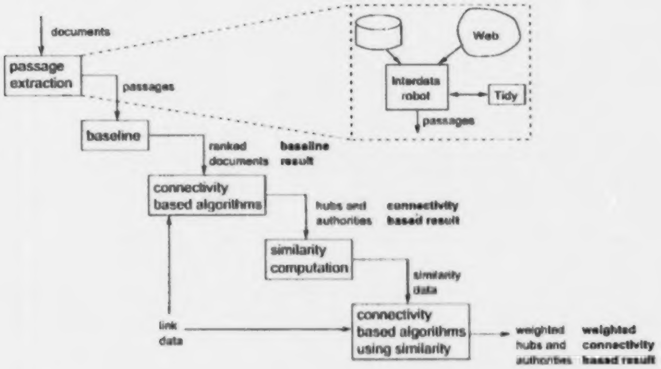

documents
passage extraction
passages
Interdata robot
Web
Tidy
passages
baseline
ranked documents
baseline result
connectivity based algorithms
hubs and authorities
connectivity based result
similarity computation
similarity data
link data
connectivity based algorithms using similarity
weighted hubs and authorities
weighted connectivity based result

Figure 1: The experimental process. Bold text refers to the submitted runs.

Baseline. First 10 passages – title and paragraphs – are extracted from each document and indexed using a stop-list augmented with Web stopwords, the Porter's stemming algorithm, and by keeping non-stemmed words; for example, the word "White" has been stored together with "white". Title-only and title-description queries are automatically generated, and indexed as passages did. For each query, top 10,000 passages are retrieved and ranked by F-4 [2]. The lists of retrieved passages are reweighted through blind relevance feedback by considering top 100 passages as relevant. The lists of newly 10,000 retrieved passages are mapped to retrieved documents. The document score is the sum is the of the scores of the mapped passages.

Connectivity-based algorithm. A modified version of the HITS (Hyperlink Induced Topic Search) algorithm is applied on the provided link files, where the link weight is the baseline score;

Similarity-based algorithm. In- and out-links are weighed using similarity among documents; the similarity between two documents is the average similarity between the passages of a document and the passages of another document;

Connectivity-based algorithm using similarity. The modified HITS algorithm is applied on the weighted link files, where the weight of the link between two documents is the content similarity between the documents.

We have then submitted six runs – three runs for each query type, i.e. topic title-based queries and topic description and title-based queries:

- baseline: F-4-based passage ranking and query term reweighing using blind relevance feedback (PuShortBase, PuLongBase);

BEST COPY AVAILABLE

- modified HITS: baseline lists are re-ranked using authority weights that are computed considering links equally (PuShortAuth, PuLongAuth);
- modified HITS with weighted links: baseline lists are re-ranked using authority weights that are computed weighing links by text similarity (PuShortWAuth, PuLongWAuth).

1.1 Web stopword list

Stoplists are fundamental tools to reach effective and efficient indexing and retrieval results. So far, different stoplists have been developed for different languages and application domains. Differently from classical document collection, the Web is a potentially infinite universe which is about many different subjects and is a container of many different languages. Thus, a search engine should be provided with many different stoplists to consider such a myriad. However, a word, which is a stopword in a stoplist of an application domain or for a language, could be a keyword within another application domain or for another language.

Web pages are often rich of terms, words or sentences including strings that represents words of languages and protocols of the Internet and of the Web. Actually, Web pages are written using a mark-up language, such as HTML or XML. Therefore, these sort of documents contain both text encoding the information that are explicitly communicated to the user, and text representing "net-stopwords", i.e. mark-up language or Internet words being used to write the page down and to allow for the transmission of the page through networks. Indexing algorithms does sometimes extract "net-stopwords" and have to decide if to keep them as keywords.

To address the problem of the presence of "net-stopwords", we have developed a list of 65 stopwords that are considered very frequent in Web pages, and that can be considered as "net-stopwords". Examples of "net-stopwords" are HTML words, such as "www" or "html", or the most common strings that are used to compose electronic mail or Web addresses, such as "com" or "net".

We computed the frequency distribution of the most used words in the training set. We realized that the classical stoplist is still valid for IR applications on the Web. Furthermore, we identified additional words and we selected very frequent words that are about the World Wide Web and not about a specific domain.

1.2 Passage Retrieval

We used passage retrieval because Web pages are often long or multi-topic documents. Using the mark-up information and some numerical parameters, such as passage size, we have extracted passages from the Web pages and have used these passages as source of evidence to index and retrieve documents. From each document, we have extracted the following passages:

BEST COPY AVAILABLE

- meta-data fields, such as authors' names, keywords, and description, identified by the `<META>` tag,
- page title, identified by the `<TITLE>` tag,
- paragraphs, identified by the `<P>` tag,
- headings, identified by the `<H1>`, `<H2>`, and `<H3>` tags.

We have chosen these tags assuming these passages are likely to include most part of the discriminating keywords. Moreover, we assumed that some tags play a specific role to carry the semantic content description of Web pages; for example, page authors are likely to use headings to give weight to the keywords being stored in the headed passages. Similarly, we assumed that page title often contains important keywords, and that paragraphs are effective ways to structure the relevant information. We also assumed that meta-data are effective means to represent relevant information, and to identify relevant document. Indeed, meta-data, e.g. like keywords and description, are manually filled by the page's authors and then they are likely to describe the semantic content precisely and exhaustively. However, we realized that a very low percentage of documents include manually filled meta-data that are the result of an intellectual work of content description, while many of the documents with meta-data include automatically filled fields, such as the page In total, we have extracted 8.6 million passages. The engine retrieved and scored passages before building the list of retrieved documents. composer product name, that are poor semantic content descriptors.

The formula $g_P(d) = \sum_{i=1}^{N_d} g(d_i)$ has been used to compute the document score starting from the passage scores, where: N_d is the number of passages $d_i, i = 1, ..., N_d$ of d, $g(d_i) = \sum_{k=1}^{K} q_k \; t_{ik} \; c_{1k}$, K is the number of index terms, $q_k = 1$ if index term k occurs in the query, $t_{ik} = 1$ if index term k occurs in d_i, c_i is the relevance term weight computed using the distribution of terms in passages.

1.3 Connectivity-based Algorithms

We employed the HITS (Hyper-link Induced Topic Search) algorithm [4] to re-rank the baseline document list. The document list has been given as input to the algorithm and each document has been assigned an authority and a hub weights. Authority weights has been used to rank the list, so that the most authoritative pages are placed on the top of the list.

1.4 Similarity-based Link Weighing

HITS and the modified version we used in our experiments ignore the semantic content of the linked documents, then, links between documents with a dissimilar content are treated equally to links between documents about similar

BEST COPY AVAILABLE

content. To test if semantic content affects the effectiveness of connectivity-based algorithms, we weigh the links being provided with the test collection using a similarity function.

Inter-document similarity-based reweighing is computed as follows. We are provided with two link files – in-link and out-link files. Given a link file, a new weighted link file is computed. After weighing link files, we obtain two weighted link files being similar to the provided link files, but links are weighted using a linear combination of the manual weight, and the similarity between the linked documents. This linear combination uses the coefficient α.

The weight of the link between d and c is $\alpha + (1-\alpha)sim(d,c)$, where α is the weight given to the manual link and $sim(d,c)$ is the inter-document similarity between d and c ($\alpha = 0.5$, in the submitted runs). Figure 2 depicts an example of combination of Web links and similarity links; for example, the weight of the Web link from A to B is $\alpha + (1-\alpha)\frac{1}{3}(.5 + .4 + .2)$ which is the average passage similarity link weight.

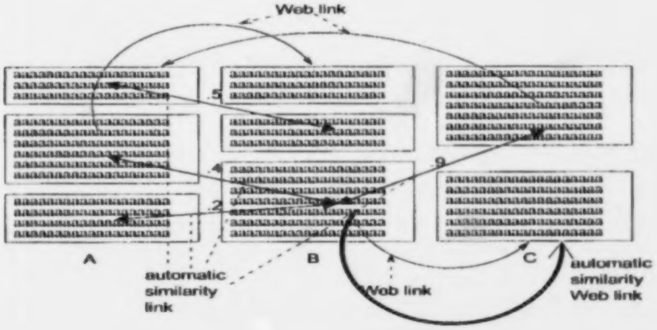


Figure 2: An example of combination of Web link and similarity link. Light arrows represent Web links starting from a passage and ending to a page. Heavy arrows represent similarity link between passages.

2 Development Approach

We have chosen to implement mainly in-house the software being necessary to carry experiments out. We have preferred to supervise the underlying algorithms and to make changes to the software whenever it was necessary.

As regards to the step of passage extraction, we developed a tool to extract passages from Web pages. The tool was originally been designed as a software agent that follows the Web links to retrieve the Web pages; indeed, it is a robot. This robot has been developed within the National InterData research project [5]. For the purposes of the TREC experiments, a different version of the robot has been designed and developed because the data to be retrieved were locally

BEST COPY AVAILABLE

stored, and not on the Web. Moreover, the data are encoded in SGML also and then the tool has been modified to deal with this additional format. To only extract the tagged text, our robot employed a tool for HTML syntax analysis, called Tidy, that is reported in [6]. Tidy allows for correcting HTML syntax by adding, for example, missing end tags.

We reused the TACHIR software library to implement the indexing and retrieval engine [7]. The indexing, retrieval and connectivity analysis software has entirely been implemented in C++ and persistence has been managed using GNU Database Manager (GDBM) [8].

3 Experimental Hypotheses and Questions

In carrying our experiments out, we have made some hypothesis, which are listed in the following:

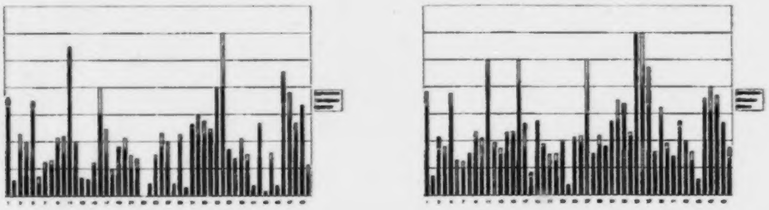
- Passage retrieval and blind relevance feedback are useful. Past research and experiments have shown that extracting passages and using blind relevance feedback are effective means to improve performance. We have therefore employed those methods and produced baseline results that already incorporates them. Thus, we made no comparison with experiments without passage retrieval and blind relevance feedback.

  At training phase, we tested that passage retrieval and blind relevance feedback for query term reweighing are effective means to improve performance. Training was performed using the WT2G test collection and the TREC-7 and TREC-8 topic sets. We then decided to use passage retrieval and blind relevance feedback as method to produce the baseline results.

- Only a part of a Web page can be indexed to reach acceptable levels of effectiveness. We assumed that the representation of relevant information are concentrated in few passages and few passage types. Specifically, we assumed that we could concentrate indexing on only some tags (some tags are useful, others are useless), only the top part of the document, only the initial part of passages.

- The documents are written using the Latin alphabet and in English. We have therefore developed no software being dependent to specific alphabet or language. Apart the Web stoplist, only an English stoplist has been used and only the Porter's

Our experiments aimed to test the impact on effectiveness of connectivity-based algorithms, similarity-based link weighing and connectivity-based algorithms. Specifically, we wanted to test whether the use of the modified version of the HITS algorithm increases the levels of effectiveness reached through the baseline results. Moreover, we wanted to test whether weighing the links employed to perform the modified version of the HITS algorithm increases the levels of effectiveness reached through the baseline results. In other words, we tested whether adding information about the semantic content of the linked documents is useful.



(a) (b)

Figure 3: Baseline results using topic title-based (a) and description-based queries (b)

4 Official Results

Figures 3(a) and 3(b) depict the results reached through the baseline methods using the short query and the long query version, i.e. queries being based on the topic title only, and queries being based on the topic title and description. In both Figures, we reported the best, the median and our precision level after 100 retrieved documents for each topic. Each bar of a histogram refers to a topic and depicts the proportion of a precision level – best, median and our – with respect to the percentage of documents being relevant to the topic. The grey (bottom) part of a bar refers to the best result, the dark (middle) part of a bar refers to the median result, the light (top) part of a bar refers to our result. Table 1 reports the official results expressed as average R-Precision (precision after R docs retrieved).

		Unweighted		Similarity-based	
	Baseline	Authorities	Hubs	Authorities	Hubs
Title-only	18.2%	18.9%	18.9%	18.9%	18.9%
Title+description	16.7%	11.7%	16.7%	16.7%	16.7%

Table 1: The official results.

On average, our results are worse than the median results. In some cases, our result is far less than the median, and then of the best result. Note that, in some cases, our result is comparable to or better than the median or to the best result.

The results reached using topic title and description-based queries are comparable to those reached using topic title-based queries. Indeed, no significant improvements have been reached using longer queries. On average, long query results are better than short query results.

The results reached using the connectivity-based algorithms – modified HITS and similarity weighing links – give no significant variations of the baseline results. The pictorial description of those results would be very similar, and would be equal for many topics.

5 Problems

As we have participated to TREC at the first time, we encountered plenty of problems, mainly because of the need of interleaving implementation issues and methodological problems. This meant that we had to sacrifice some methodological solutions to finish the experiments on time and to cope with some implementation deficiencies. As consequence, we had to limit: The number of passage types – we used only meta-data, paragraphs and headings; the number of retrieved passages – only 10,000 passages are retrieved for each query; the number of passage words – we considered 20 words per passage only; the use of query expansion – queries were not expanded after blind relevance feedback.

Moreover, we had implementation problems. We think that W3C HTML Tidy is too "severe", yet is a useful and powerful tool to extract passages from Web pages. We encountered other problems related to the presence of Web pages written in Japanese that created some difficulties for our passage extraction software. We had to eliminate these documents semi-automatically. We have "lost" some pages because of the presence of frames and CGI script calls. In one case, we found a page being splitted into two parts – a part is read by the browser if it is enable to process frames, otherwise, the browser reads the other part. The part that is activated if the is enable to process frames stores a call to a CGI scripts and no other data is stored. The other part stores the text that has been indexed, but that is different from the text that would be produced by the CGI script, if called. Therefore, our software indexed the "explicit" text, by the judges maybe assessed the text being produced by the CGI script. We encountered some problems in dealing with passage extraction from very long and non-tagged texts, such as those included by `<PRE>` tags. Of course, we were unable to cope with "wrong" query words, such as "nativityscenes".

6 Conclusions

After this first experience, we learned a lot about basic issues of text retrieval and about advanced issues of Web page retrieval. Basically, we learned that investing human resources is the most crucial factor affecting results. We believe that we can invest more time to the methodological issues at the next TREC because many implementation problems have been addressed at this TREC.

BEST COPY AVAILABLE

It is necessary to index all the document – all the tags because they are very often used for presentation purpose and not for carrying semantics; this means that, for example, headings carry no more information than other pieces of text. All the of the document parts because a document can be relevant because there can be a relevant passage on the bottom; this is the case of long documents, especially, but also for short and structured documents, such as list of items that include links. All the passage because there can be many long passages that store relevant information in the middle or at the end of the text.

Passage retrieval requires too large data files if implementing passages as individual documents. We had then to cut passages off, but we have lost many useful information; alternative data structures that employ proximity-based collocations are currently under investigation. More sophisticated document scoring system is necessary. Summing passage scores is a rather simplistic way to compute the score of the document which passages belong to. There can be irrelevant very large documents with many short high scored small passages or few high scored large passages.

The connectivity-based experiments gave no variations probably because we applied no expansion of the root page set by adding in-linked and out-linked pages. Thus, the root page set was equal to the base page set and the connectivity-based algorithms have made no significant changes to the original ranking. The use of the baseline document score and of the similarity-based link weight gave no contributions.

Our experiments confirmed that in classical IR, documents are organized texts and text organization carries some semantics about the document content; on the contrary, Web documents are sometimes more structured than classical documents, but this structure carries little semantics about the document content. In classical IR, end users are expert persons about the application domain, then queries are well formulated and often the query vocabulary correspond to the vocabulary used by the document authors. On the contrary, Web queries are formulated by non-expert persons because the Web collection are not about an application domain.

References

[1] E. Voorhees and D. Harman, editors. *Overview of the Sixth Text Retrieval Conference (TREC-6)*, volume 36(1), 2000.

[2] S.E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129–146, May 1976.

[3] M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[4] J. Kleinberg. Authorative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604-632, September 1999.

[5] F. Crivellari and M. Melucci. Awir: Prototipo di un motore di ricerca per la raccolta, indicizzazione e recupero di documenti web sulla base dei loro frammenti. Rapporto tecnico T2-S12, Progetto INTERDATA - MURST e Università

di Padova: "Metodologie e tecnologie per la gestione di dati e processi su reti Internet e Intranet". Tema 2: "Estrazione di informazioni distribuite sul WWW"., ftp://ftp-db.deis.unibo.it/pub/interdata/tema2/T2-S12.ps, Febbraio 1999. In Italian.

[6] World Wide Web Consortium (W3C) HTML Tidy. `http://www.w3.org/People/Raggett/tidy/`, October 2000. Last visited: October 25th, 2000.

[7] M. Agosti, F. Crestani, and M. Melucci. Design and implementation of a tool for the automatic construction of hypertexts for Information Retrieval. *Information Processing & Management*, 32(4):459–476, July 1996.

[8] GNU Database Manager (GDBM). `http://www.gnu.org/software/gdbm/gdbm.html`, October 2000. Last visited: October 25th, 2000.

University of Sheffield TREC-9 Q & A System

Sam Scott* and Robert Gaizauskas
`{s.scott,r.gaizauskas}@dcs.shef.ac.uk`

Department of Computer Science
University of Sheffield
Regent Court, Portobello Road
Sheffield S1 4DP UK

1 Introduction

The system entered by the University of Sheffield in the question answering track of TREC-9 represents a significant development over the Sheffield system entered for TREC-8 [6] and, satisfyingly, achieved significantly better results on a significantly harder test set. Nevertheless, the underlying architecture and many of the lower level components remained the same. The essence of the approach is to pass the question to an information retrieval (IR) system which uses it as a query to do passage retrieval against the test collection. The top ranked passages output from the IR system are then passed to a modified information extraction (IE) system. Syntactic and semantic analysis of these passages, along with the question, is carried out to identify the "sought entity" from the question and to score potential matches for this sought entity in each of the retrieved passages. The five highest scoring matches become the system's response.

2 System Description

2.1 Overview

The key features of the system setup, as it processes a single question, are shown in Figure 1. Firstly, the (indexed) TREC document collection and the question are passed to an IR system which treats the question as a query and returns top ranked passages from the collection. As the IR system we used the Okapi system [8][1]. Following this, the top ranked passages are run through a text filter to remove certain text formatting features which cause problems for downstream components. Finally, the question itself and the filtered top ranked passages are processed by a modified version of the LaSIE information extraction system [5], which we refer to below as QA-LaSIE. This yields a set of top ranked answers which are the system's overall output.

The reasoning behind this choice of architecture is straightforward. The IE system can perform detailed linguistic analysis, but is quite slow and could not process the entire TREC collection for each query, or even realistically pre-process it in advance to allow for reasonable question

---

*Current address: Institute for Interdisciplinary Studies, Carleton University, Ottawa, K1S 5B6, Canada. `sscott@ccs.carleton.ca`.

[1] Software available at: `http://dotty.is.city.ac.uk/okapi-pack/`. For TREC-8 we both used the NIST-supplied top documents and passages from UMass's INQUERY system [2] which UMass kindly provided for us. Our switch from INQUERY to Okapi was prompted by the acquisition of Okapi and Okapi expertise in-house.

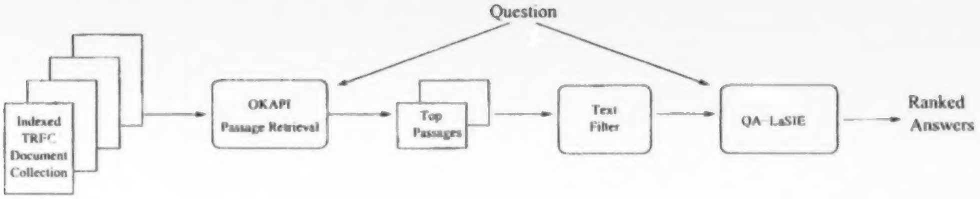


Figure 1: System Setup for the Q & A Task

answering performance during the test run. IR systems on the other hand are designed to process huge amounts of data. By using an IR system as a filter to an IE system we hope to benefit from the respective strengths of each.

In the next section we describe how we have parameterised Okapi for the QA task. The following section briefly describes the base LaSIE system and the succeeding section describes how it has been modified and extended to yield QA-LaSIE. Of the text filtering module we say no more, as it is of little intrinsic interest and was simply a convenient way of avoiding modifications to various components in LaSIE to deal with idiosyncrasies in the TREC collection texts.

2.2 Okapi

The Okapi IR system is based on the probabilistic retrieval model [7]. Since we used the system "off the shelf", we discuss here only the parameterisation adopted for the QA task, and not the underlying model. Aside from using a slightly modified version of the stop list provided with Okapi, no parameterisation of the indexing process took place. However, to utilise the passage retrieval capabilities of Okapi a number of parameters need to be set for the searching process. We set the minimum number of paragraphs to be returned to 1, the maximum number to 3, and the paragraph step unit to 1 (this parameter determines how much the sliding passage window moves between comparisons in the passage ranking process). Determining that the maximum number of paragraphs per passage should be 3 was a matter of some experimentation, and it interacted with the decision about how many passages to select per question (because of processing times in the IE system).

Our experimentation was carried out using the TREC-8 QA track question set, but the TREC-9 document collection (the latter is a superset of the TREC-8 collection). Looking at the top 5, 10 and 20 documents returned for each question we discovered that these document sets contained answers for 160, 175 and 184 of the 198 TREC-8 questions respectively. We then tried experimenting with passage retrieval using the Okapi default settings for minimum passage length (1 paragraph) and maximum passage length (20 paragraphs), and keeping the best passage only if the retrieval engine's score for it was higher than for the entire document. For 158 of the questions this resulted in an answer being found in the top 5 passages – a loss of only 2 over the full document approach. Deciding, therefore, that passage retrieval was worthwhile, we experimented with the maximum passage length parameter. By reducing it to 3 paragraphs, and always preferring best passages to full documents, even if the full document score was higher, we discovered no answers were lost (i.e. the top 5 best passages per question still contained answers for 158 of the questions). Furthermore, running the passages of maximum length 3 through the QA-LaSIE system led to considerably higher mean reciprocal rank (MRR scores) than using the full documents, presumably because there were fewer distractors.

Given the significantly smaller amount of text to be processed by QA-LaSIE using passages of at most 3 paragraphs, we were encouraged to examine more top passages per question. By considering the top 20 best passages we discovered that 164 questions had answers in the retrieved passage sets; and the MRR scores of QA-LaSIE against these were higher than for the top 5 best passages per question. These then were the parameters we finally settled upon for the TREC-9 evaluation: top 20 passages per question with maximum passage length 3 paragraphs.

2.3 LaSIE

The LaSIE system used to perform detailed question and text analysis is largely unchanged in architecture from the IE system as entered in the last Message Understanding Conference evaluation (MUC-7) evaluation [5]. The principal components of the LaSIE system are the first eight modules shown in Figure 2 as executed interactively through the GATE Graphical Interface [3]. The system is essentially a pipeline of modules each of which processes the entire text before the next is invoked. The following is a brief description of each of the component modules in the system:

Tokenizer Identifies token boundaries (as byte offsets into the text) and text section boundaries (text header, text body and any sections to be excluded from processing).

Gazetteer Lookup Identifies single and multi-word matches against multiple domain specific full name (locations, organisations, etc.) and keyword (company designators, person first names, etc.) lists, and tags matching phrases with appropriate name categories.

Sentence Splitter Identifies sentence boundaries in the text body.

Brill Tagger [1] Assigns one of the 48 Penn TreeBank part-of-speech tags to each token in the text.

Tagged Morph Simple morphological analysis to identify the root form and inflectional suffix for tokens which have been tagged as noun or verb.

Parser Performs two pass bottom-up chart parsing, pass one with a special named entity grammar, and pass two with a general phrasal grammar. A 'best parse' is then selected, which may be only a partial parse, and a predicate-argument representation, or quasi-logical form (QLF), of each sentence is constructed compositionally.

Name Matcher Matches variants of named entities across the text.

Discourse Interpreter Adds the QLF representation to a semantic net, which encodes the system's background world and domain knowledge as a hierarchy of concepts. Additional information inferred from the input using this background knowledge is also added to the model, and coreference resolution is attempted between instances mentioned in the text, producing an updated discourse model.

For standard IE template filling tasks, a final Template Writer module reads the discourse model produced by the Discourse Interpreter, derives template slots fills and writes out the filled templates.

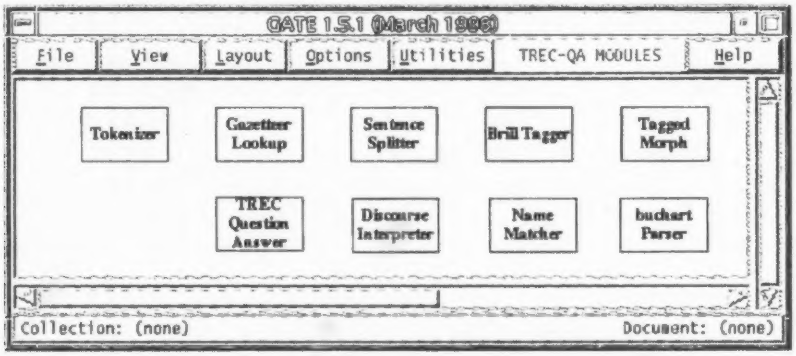
2.4 QA-LaSIE

The QA-LaSIE system takes a question and a set of passages delivered by the IR system and returns a ranked list of proposed answers for the question. Figure 2 shows the end-to-end QA-LaSIE system as entered in TREC-9. Four key adaptations have been made to move from the base IE system described in the previous section in a system capable of carrying out the QA task:

1. a specialised grammar was developed to analyse questions and added to the parser;
2. the discourse interpreter was modified to allow the QLF representation of each question to be matched against the discourse model of a candidate answer text, using the coreference mechanism;

3. an answer identification procedure which scored all discourse entities in each candidate text as potential answers was added to the discourse interpreter;
4. a TREC Question Answer module was added to examine the discourse entity scores across all passages, determine the top 5, and then output the appropriate answer text.

Further detailing these alterations will not communicate the essence of the QA-LaSIE approach to question answering. Therefore, in the following subsections we describe the key processes involved in the QA-LaSIE approach to question answering, including, where necessary, further information about the general approach to text processing taken in LaSIE.



Collection: (none)
Document: (none)

Figure 2: QA-LaSIE System Modules

2.4.1 Parsing: Syntactic and Semantic Interpretation

In the LaSIE approach, both candidate answer passages and questions are parsed using a unification-based feature structure grammar. As input the parser receives one sentence at a time and along with the original words of that sentence also receives: a part-of-speech tag from the Penn tagset for each word, morphological information for each noun and verb (word root plus affix), and zero or more phrases tagged as named entities. As output the parser produces as representation of the sentence in a "quasi-logical form" – a predicate-argument representation that stands somewhere between the surface form of the sentence and a fully interpreted semantic representation in a standard logical language. In particular the QLF representation defers issues of quantifier scoping and of word sense disambiguation.

To take a simple example, the sentence fragment *Morris testified that he released the worm ...* is parsed and transduced to the representation

```
person(e1), name(e1,'Morris'), gender(e1,masc), uncertain(e1,person),
testify(e2), time(e2,past), aspect(e2,simple), voice(e2,active),
lsubj(e2,e1),
release(e3), time(e3,past), aspect(e3,simple), voice(e3,active),
pronoun(e4,he),
lsubj(e3,e4)
worm(e5), number(e5,sing), det(e5,the),
lobj(e3,e5),
proposition(e6),
main_event(e6,e3),
lobj(e2,e6)
```

BEST COPY AVAILABLE

The name information is derived from the Gazetteer lookup stage (where *Morris* is recorded as a male first name), the tense information from the morphological analysis stage, and the grammatical role information from annotations on context-free rules in the grammar. In this case these rules encode that in English sentences which consist of a noun phrase followed by a verb phrase, which in turn consists of a verb in the active voice and a sentential complement, the noun phrase prior to the verb is the subject and the sentence following it is the object. For common nouns and verbs, the lexical root of the word becomes a predicate in the QLF language.

Both noun phrase heads and verb group heads are given unique discourse entity references of the form e_n. This allows modification relations (e.g. of prepositional phrases) or grammatical role information (e.g. subject and object relations) to be captured via binary predicates holding of these entities. In cases where parsing fails to capture all this information (e.g. when only simple noun phrase, verb group, prepositional phrase or relative clause chunks are found and not a spanning parse for the sentence) then partial QLF information can be returned, making the system robust in the face of grammatical incompleteness.

Each sentence in a candidate answer passage is analysed in this fashion. In addition so is the question, using a special question grammar. This grammar produces a QLF for the question in much the same style as above. For example, a question such as *Who released the internet worm in the late 1980s?* would be analysed as:

```
qvar(e1), qattr(e1,name), person(e1),
release(e2), time(e2,past), aspect(e2,simple), voice(e2,active), qcon(e2,verb),
lsubj(e2,e1),
worm(e3), number(e3,sing), det(e3,the),
lobj(e2,e3),
name(e4,'Internet'), qual(e3,e4)
```

Note the use of the special predicate, `qvar` (question variable), to indicate the 'entity' requested by the question. In this case the `qvar` can also be typed because *who* tells us the entity of concern is a person, and we presume (by encoding this in the transduction rules) that the attribute we are seeking here is a name (and not, e.g., a definite description such as *a guy at MIT*). In other cases where the interrogative pronoun is more generic (e.g. *what*) the type of the `qvar` and the attribute sought of it may not be so readily determinable.

2.4.2 Discourse Interpretation of Candidate Answer Texts

Once a text has been parsed and each sentence has been assigned a QLF representation as discussed in the preceding section, the next component of QA-LaSIE, the discourse interpreter, integrates the texts into a discourse model. The discourse model is a specialisation of a semantic net which supplies the system's background domain knowledge. For IE applications, this domain-specific background knowledge assists in extraction tasks, by allowing template slot values to be inferred from it together with information supplied in the text being analyzed. However, for the TREC QA task there is no specific domain, and so this role of the semantic net is not relevant (though a very basic "generic" world model is employed).

The real function of the semantic net in the QA task is to provide a framework for integrating information from multiple sentences in the input. As the QLF representation of each sentence is received by the discourse interpreter, each entity is added as an instance node in the semantic net associated with its type node (the single unary predicate in which it occurs) – e.g. given `worm(e5)`, `e5` is linked to the `worm` node in the net, if it already exists, and to a new node labelled `worm` if not. Added to each such entity node is an attribute-value structure, or property list, containing all the attribute and relational information for this entity (all the binary predicates in which it occurs in the input).

In addition to adding a sentence's QLF to the semantic net in this fashion, one further node is added representing the sentence itself. This sentence entity has a sequence number indicating the sentence's position in the text, and also has an attribute recording the entity numbers of every entity occurring in the text. Thus, the discourse model aims to model not only the content of the discourse, but simple aspects of the discourse structure itself.

After each sentence has been added to the discourse model, the main task of the discourse interpreter commences. This is to determine coreference relations between entities in the current sentence and entities already added to the model from previous sentences in the input. There is not space to detail this algorithm here (see [4]), but in essence it relies upon factors including the semantic type compatibility, attribute compatibility, and textual proximity of potential coreferents. Once a coreference has been established between two entities, the two are merged by replacing all references to the two entity numbers by references to just one of them. However, the surface realisations which initially served as triggers for the creation of each distinct entity node are retained as attributes of the merged entity, and can be used later, e.g. to generate a text string as an answer.

2.4.3 Answer Identification

Given that a discourse model for a candidate answer passage has been constructed as just described, the QLF of the question is added to this model and treated as sentence 0. The coreference procedure is run and as many coreferences as possible are established between entities in the question and those in the passage[2].

In the TREC-8 version of QA-LaSIE this procedure was the primary question answering mechanism: if the qvar was resolved with an entity in the text then this entity became the answer; if not, then no answer was proposed. This approach had several major drawbacks. First, it permitted only one answer per question, whereas the QA track allows up to five answers to be proposed. Second, it was very fragile, as coreference tends to be difficult to establish.

Given these weaknesses, the TREC-9 system follows a significantly different approach. Instead of attempting to directly corefer the qvar with an entity in the text, entities in the text are scored in a way which attempts to value their likelihood as answers. The best scores are then used to select the answers to be returned from the passage.

The details of this approach are as follows. The discourse model is transversed twice, sentence by sentence:

1. *Sentence Scoring* On the first pass, the sentences are given an integer score. The entities in the question are interpreted as "constraints" and each sentence in the answer text gets one point for each constraint it contains. This has the effect that sentences which contain entities that have been detected as coreferring with entities in the question will be rewarded. Typically this will be sentences which contain named entities mentioned in the question, or sentences which have definite noun phrases or pronouns which have already been resolved (as part of discourse interpretation of the passage).

2. *Entity Scoring* On the second pass, the system looks in each sentence for the best possible answer entity. To be considered a possible answer, an entity must be an object (not an event), and must not be one of the "constraints" from the previous step. If the qvar has a tt qattr (see 2.4.1 above), then the entity must also have the specified attribute to be considered a possible answer. The entities in a given sentence are compared to the qvar and scored for semantic similarity, property similarity, and for object and event relations.

---

[2] The standard coreference procedure uses a distance metric to prefer closer to more distant potential coreferences. Clearly this is irrelevant for questions which are not part of the original text. Hence we have switched off the distance-preference heuristic for coreference in this case.

Semantic and property similarity scores are determined as for generic coreferencing. A semantic similarity score between 0 and 1 is computed, depending on how closely semantically related two things are. For instance, if the `qvar` has the type `person`, then an entity that also has type `person` will receive a semantic similarity of 1. In general, the semantic similarity is related to the inverse of the path length that links the two semantic types in the ontology. If the two semantic types are on different branches of the hierarchy, the score is 0.

The property similarity score is also between 0 and 1 and is a measure of how many properties the two instances share in common and how similar the properties are.

The object and event relations scores were motivated by failure analysis on the original system and were tuned through test runs. The object relation score adds 0.25 to an entity's score if it is related to a constraint within the sentence by apposition, a qualifying relationship, or with the prepositions *of* or *in*. So if the question was *Who was the leader of the teamsters?*, and a sentence contained the sequence … *Jimmy Hoffa, Leader of the Teamsters,* … then the entity corresponding to *Jimmy Hoffa* would get the object relation credit for being apposed to *Leader of the Teamsters*.

The event relations score adds 0.5 to an entity's score if:

(a) there is an event entity in the QLF of the question which is related to the `qvar` by a `lsubj` or `lobj` relation and is not the `be` event (i.e. derived from a copula construction), and

(b) the entity being scored stands in the same relation (`lobj` or `lsubj`) to an event entity of the same type as the `qvar` does. So if the question was, *What was smoked by Sherlock Holmes?* and the answer sentence was *Sherlock Holmes smoked a pipe*, then the entity *a pipe* would get the event relations credit for being in the `lobj` relation to the verb *to smoke*.

This represents a significant weakening of the requirement in our TREC-8 system that the `qvar` had to match with an entity in the answer text which stood in the same relation to its main verb as the `qvar` did with the main verb in the question, as well the main verbs and other complements being compatible. Here a bonus is awarded if this the case; there it was mandatory.

Finally, the entity score is normalized to bring it into the range [0,1]. This is motivated by the idea that if two sentences have equal scores from step 1. above, the entity score should break the tie between the two, but should not increase their scores to be higher than a sentence that had a better score from step 1. Normalizing the score improved performance slightly in tests on the Trec 8 questions.

3. *The Total Score* For every sentence, the "best" answer entity is chosen according to the Entity Scoring as described above. The sentence and entity scores are then added together and normalized by dividing by the number of entities in the question plus 1 The sentence instance is annotated to include the total score, the best entity (if one was found), and the "exact answer". The exact answer will be the `name` of the best entity if one was identified during parsing. Otherwise this property is not asserted.

2.4.4 Answer Output

The answer output procedure gathers the sentence total scores, as described in the preceding section, from each sentence in each of the passages analyzed by QA-LaSIE, sorts them into a single ranking, and outputs answers from the overall five highest scoring sentences.

We submitted four runs to the TREC-9 evaluation, two in the 50-byte category and two in the 250 category. These four runs are explained below:

System	Run	Mean Reciprocal Rank	Correct Answers	Rank in Class
shef-trec8	50	.081	N/A	15/17
okapi-baseline	50	.157	N/A	14/17
shef50ea	50	.329	89/164	4/17
shef50	50	.368	98/164	3/17
shef-trec8	250	.111	N/A	22/24
okapi-baseline	250	.395	N/A	11/24
shef250	250	.490	127/164	4/24
shef250p	250	.506	130/164	4/24

Table 1: Development Results on TREC-8 Questions

shef50ea This is the "exact answer" run, submitted in the 50-byte category. If a high scoring sentence was annotated with a `trec9_exact_answer` attribute then this is assumed to be the answer. If there is no "exact answer", then the code looks for a `trec9_answer_entity` and outputs the longest realization of that entity as the answer. If there is no "answer entity", which can happen occasionally, then a default string is output. In all cases, the string is trimmed to 50 bytes if necessary, by trimming characters from the left hand side.

shef50 For this run, the system looks for the first occurrence of the `trec9_answer_entity` in the sentence and then outputs 50 bytes of the sentence centered around that entity. The 50-bytes will never go outside of the answer sentence (if the first occurrence is the first word, then the 50 bytes will be the first 50 bytes of the sentence, and so on). If the sentence is shorter than 50 bytes, then the full sentence is output as the answer. If there is no answer entity, the middle 50 bytes are output.

shef250 Same as shef50, but up to 250-bytes or the full sentence is output (whichever is shorter).

shef250p For this run, the answer for shef250 is computed, then the answer is padded to 250 bytes if necessary by adding characters from the file to both ends, going outside the confines of the sentence if necessary.

3 Results

In the next two sections we describe results obtained from the system, first during development, and then in the TREC-9 test run.

3.1 Development Results

Our development setup consisted of the 200 TREC-8 questions, the TREC-9 document collection (a superset of the TREC-8 collection), and Perl patterns for identifying correct answers for the TREC-8 questions in proposed answer strings, made available by Ellen Voorhees following the TREC-8 evaluation. These resources allowed us to modify our system, re-run against the 198 TREC-8 questions and score our results in a tight modify-evaluate loop.

One initial baseline experiment was to see if QA-LaSIE was actually adding value over a naive approach that simply used Okapi passage retrieval with a maximum passage length of one paragraph and then trimmed this paragraph to 50 or 250 bytes. Taking the top 5 one paragraph passages for each query in the development set and trimming them to the central 50 or 250 bytes led to MRR scores of 0.157 for the 50 byte responses and .395 for the 250 byte responses. This totally naive approach would have placed 14-th of 17 in the TREC-8 50-byte system ranking and

System	Run	Mean Reciprocal Rank		% Correct Answers in Top 5		Rank in Class
		Strict	Lenient	Strict	Lenient	
shef50ea	50	.159	.172	23.6	25.7	28/35
shef50	50	.206	.217	31.1	32.1	21/35
median (of 35)	50	.227				
mean (of 35)	50	.220	.227	31.0	32.2	
shef250	250	.330	.343	48.5	49.4	28/43
shef250p	250	.345	.357	50.9	51.3	23/43
median (of 43)	250	.349				
mean (of 43)	250	.351	.363	49.0	50.5	

Table 2: TREC-9 Results

joint 11-th of 24 in the 250-byte system ranking. In both cases these results were considerably higher than our own entries in TREC-8.

Thus, we started with a sobering baseline to contend with. However, following development of the new approach described above in section 2.4.3 and numerous experiments with various parameter settings we arrived at the best development results presented in Table 1. For comparison the Sheffield results from TREC-8 and the Okapi baseline experiment results are also included in this table.

3.2 Final Evaluation Results

Mean reciprocal rank scores for the four Sheffield runs are shown in Table 2, for both lenient and strict scorings. We have also computed the percentage of questions for which a correct answer was present in the top 5 answers returned by system. The final column shows the system's rank in the various answer categories, with respect to the number of participating systems (this is calculated with respect to the strict scoring of mean reciprocal rank). Also included are the median and mean scores for systems participating in each category. From these figures it can be seen that in both 50 and 250 byte categories the better Sheffield system is close to, but just slightly below, the median and mean (shef50 is third highest below median, and shef250p second highest below median).

4 Discussion

At this point we do not have the information to allow us to apportion faults between Okapi and QA-LaSIE. In training on the TREC-8 questions (but against the TREC-9 document collection) Okapi was returning answer-containing passages for about 83% of the questions. On this basis the best QA-LaSIE mean reciprocal rank scores obtained in development were around .37 for the 50-byte runs and just over .50 for 250-byte runs, as presented above in Table 1.

Thus the TREC-9 test results represent a significant drop with respect to training results. Nevertheless, with respect to our best TREC-8 MRR results (.081 for the 50-byte run, .111 for the 250-byte run), these figures represent a very significant improvement, especially given that the question set is significantly larger and the questions are "real", as opposed to what were artificially created back-formulations in many cases in TREC-8. And, they validate the central hypothesis of our TREC-9 work that we should abandon our previous rigid approach in which answer text entities either met constraints imposed by the question or did not, in favour of a looser approach which scored potential answer entities in terms of various factors which suggested that they might be an answer.

Acknowledgements

The authors would like to thank Hideo Joho, Department of Information Studies, University of Sheffield, for help with installing and parameterising Okapi.

References

[1] E. Brill. A simple rule-based part-of-speech tagger. In *Proceeding of the Third Conference on Applied Natural Language Processing*, pages 152–155, Trento, Italy, 1992.

[2] J.P Callan, W.B. Croft, and S.M. Harding. The INQUERY retrieval system. In *Proceedings of the 3rd International Conference on Database and Expert System Applications*, pages 78–83, 1992.

[3] R. Gaizauskas, H. Cunningham, Y. Wilks, P. Rodgers, and K. Humphreys. GATE - an Environment to Support Research and Development in Natural Language Engineering. In *Proceedings of the 8th IEEE International Conference on Tools with Artificial Intelligence (ICTAI-96)*, pages 58–66, Toulouse, France, October 1996.

[4] R. Gaizauskas and K. Humphreys. Quantitative Evaluation of Coreference Algorithms in an Information Extraction System. In S. Botley and T. McEnery, editors, *Discourse Anaphora and Anaphor Resolution*. John Benjamins, London, 2000.

[5] K. Humphreys, R. Gaizauskas, S. Azzam, C Huyck, B. Mitchell, H. Cunningham, and Y. Wilks. Description of the LaSIE-II system as used for MUC-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, 1998.

[6] K. Humphreys, R. Gaizauskas, M. Hepple, and M. Sanderson. University of Sheffield TREC-8 Q & A System. In *NIST Special Publication 500-246: The Eighth Text REtrieval Conference (TREC 8)*, 1999.

[7] S. Robertson and K. Sparck-Jones. Relevance weighting of search terms. *Journal of the American Society for Information Sciences*, 27(3):129–146, 1976.

[8] S. Robertson and S. Walker. Okapi/Keenbow at TREC-8. In *NIST Special Publication 500-246: The Eighth Text REtrieval Conference (TREC 8)*, 1999.

THE THISL SDR SYSTEM AT TREC-9

Steve Renals and Dave Abberley[*]

Department of Computer Science, University of Sheffield, Sheffield S1 4DP, UK
email: s.renals@dcs.shef.ac.uk; dca@softsound.com

ABSTRACT

This paper describes our participation in the TREC-9 Spoken Document Retrieval (SDR) track. The THISL SDR system consists of a realtime version of a hybrid connectionist/HMM large vocabulary speech recognition system and a probabilistic text retrieval system. This paper describes the configuration of the speech recognition and text retrieval systems, including segmentation and query expansion. We report our results for development tests using the TREC-8 queries, and for the TREC-9 evaluation.

1. INTRODUCTION

The TREC-9 Spoken Document Retrieval (SDR) track followed on from the TREC-8 track, using the same audio collection: 902 shows (502 hours) of US broadcast news material covering the period February–June 1998. The collection contained 21 754 individual news items, totalling 389 hours of news material. The basic task was to retrieve the set of stories relevant to each of 50 topics.

There were three principal dimensions of variation to be investigated in this year's evaluation:

Story Boundaries The main task assumed unknown story boundaries. Each episode was treated as a continuous audio stream and it was the task of the SDR system to find the location (time) of the relevant news stories. The known story boundary condition, in which stories are segmented manually and irrelevant material such as adverts are removed, was used as a contrast.

Query Length Previous SDR tracks used *short* (sentence length) queries. In TREC-9, a *terse* query was also provided for each topic, which typically contained 2–3 words, to reflect queries submitted to web search engines.

Cross-Recognizer Effects In addition to the baseline recognizer and reference (subtitle) transcripts, we also used the transcripts produced by other evaluation participants (Cambridge University and LIMSI). This illuminates the effect of speech recognizer word error rate on SDR system performance.

Much of the paper describes experiments on the development test set, using the TREC-8 SDR queries. Since that evaluation included short queries only, we generated terse queries ourselves: our TREC-8 terse queries are thus not comparable with similar queries that have been generated by other groups. In our development experiments we took average precision on the short queries as our primary metric.

The paper is structured as follows. Section 2 describes the speech recognition component of the system, which is based on the ABBOT hybrid connectionist/HMM large vocabulary speech recognizer, running in real-time mode. Section 3 outlines the text retrieval system that we have used, together with a discussion of the algorithms employed for query expansion and segmentation. Section 4 presents the results we obtained on the TREC-9 SDR track and further discussion of some of the issues raised ends the paper, along with some conclusions.

2. SPEECH RECOGNITION

2.1. Abbot

ABBOT (Robinson et al., 1996) is a connectionist/HMM system (Bourlard and Morgan, 1994) which estimates posterior phone probabilities given the acoustic data at each frame. This discriminative approach differs from that used by most recognizers in that it does not include a generative model of the data. That is, the joint probability of the acoustics and word sequence is not estimated; instead an estimate of the posterior probability of the word sequence given the acoustic data is provided. This may be interpreted as a probabilistic finite state acceptor model (Hennebert et al., 1997).

A recurrent network (RNN) trained as a phone classifier (Robinson, 1994) is used as the principal posterior probability estimator. This approach is attractive since fewer parameters are required for the connectionist model (the posterior distribution is typically less complex than the likelihood) and connectionist architectures make very few assumptions on the form of the distribution. Additionally, this approach enables the use of posterior probability based pruning in decoding (Renals and Hochberg, 1999).

---

[*] Now at: SoftSound, Cambridge CB4 0WS, UK

We produced two sets of transcriptions of the audio data for the TREC-9 evaluation, referred to as S1 and S2. Both systems used the same language model (LM) and search components. The S1 system was configured to run in realtime, while the S2 system used a richer acoustic model.

2.2. S1 Acoustic Model

The S1 acoustic model comprised two RNNs each of which estimated 54 context-independent posterior phone probabilities for each frame of acoustic data. Both networks were trained using a sequence of 12th order perceptual linear prediction features (Hermansky, 1990) (plus log energy). One network estimated the phone probabilities for the current frame conditioned on the past sequence of acoustic features. The second network was trained using a frame sequence that was reversed in time, and thus estimated the phone probabilities conditioned on the future. The two estimated probability streams were averaged in the log domain to produce a final set of probability estimates. The models were trained using the 104 hours of Broadcast News training data released in 1997 (the first half of the complete broadcast news training set).

2.3. S2 Acoustic Model

The acoustic model for the S2 system was obtained by log domain merging of the probability estimates produced by the RNNs used in the S1 system with those produced by an acoustic model using modulation-filtered spectrogram features. This is essentially the system used by the SPRACH group in the 1998 broadcast news evaluation (Cook et al., 1999; Robinson et al., 2001).

The modulation-filtered spectrogram (MSG) was developed by Kingsbury et al. (1998) as a feature representation that is robust to the signal variations caused by reverberation and noise. The robustness is obtained by emphasising modulation in the speech spectral structure occurring at rates of 16Hz or less (as measured with a critical-band-like resolution) and adapting to slowly-varying components of the speech signal (a form of automatic gain control). MSG feature processing involves first calculating an auditory-like spectrum, then filtering the amplitude in each frequency band by two parallel banks of filters, one low-pass below 16Hz, and the second bandpass between 2Hz and 16Hz. Each channel is then passed, in series, through two feedback Automatic Gain Control units with time constants of 160ms and 640ms. The resulting spectra are used as features; orthogonalization (e.g. via the discrete cosine transform) provides no benefit for these features in our experience with connectionist models. However, we do increase the robustness of the system to environmental conditions by normalizing the statistics of every feature channel to zero mean and unit variance over each segment, or over entire recordings if no segmentation is performed.

The MSG acoustic model used an MLP containing 8000 hidden units trained on the full 200 hours broadcast news training set, with the training data downsampled to 4 kHz bandwidth. Experiment has previously indicated that although the word error rate of the bandlimited MSG-based system is higher than that of the PLP-based S1 system, the errors are different and the overall performance may be improved by merging the two.

2.4. Language Modelling and Search

The same backed-off trigram LM was used by both the S1 and S2 systems (Robinson et al., 2001). Approximately 450 million words of text data were used to generate the model, comprising: the Broadcast News acoustic training transcripts (1.6M words); the 1996 Broadcast News LM text data (150M words); and the 1998 North American News text data (LA Times/Washington Post (12M words), Associated Press World Service (100M words), NY Times (190M words)). The models were trained using version 2 of the CMU-Cambridge SLM Toolkit (Clarkson and Rosenfeld, 1997) using Witten-Bell discounting. We used a lexicon containing 65 432 words, including every word in the broadcast news training data. The dictionary was constructed using phone decision tree smoothed acoustic alignments. The LM and lexicon were constructed from material pre-dating the acoustic data and were fixed throughout the evaluation.

For both systems we used a large vocabulary stack decoder CHRONOS (Robinson and Christie, 1998).

2.5. Results

Table 1 gives the word error rate estimates obtained using the S1 and S2 systems. These estimates were obtained using a 10 hour sample of the test corpus defined by NIST.

3. TEXT RETRIEVAL

3.1. Basic Text Retrieval System

We used a standard probabilistic system using a short stop list of 132 words (with an additional stop list of 78 words

System	Sub.	Del.	Ins.	WER
S1	22.0	6.1	3.9	32.0
S2	20.0	5.4	3.8	29.2

Table 1: Word error rates (WER) for the S1 and S2 speech recognition systems, estimated using a 10 hour subset of the corpus.

when processing a query), the Porter stemming algorithm and term weighting similar to that used in the Okapi system. Specifically, following Robertson and Spärck Jones (1997), we used the following function $CW(t,d)$ to compute the combined relevance weight between a term t and a document d:

$$CW(t,d) = \frac{CFW(t) * TF(t,d) * (K+1)}{K((1-b)+b * NDL(d)) + TF(t,d)}. \quad (1)$$

$TF(t,d)$ is the frequency of term t in document d, $NDL(d)$ is the normalized document length of d:

$$NDL(d) = \frac{DL(d)}{\overline{DL}}, \quad (2)$$

where $DL(d)$ is the length of document d (ie the number of unstopped terms in d). $CFW(t)$ is the collection frequency weight of term t and is defined as:

$$CFW(t) = \log\left(\frac{N}{N(t)}\right), \quad (3)$$

where N is the number of documents in the collection and $N(t)$ is the number of documents containing term t. The parameters b and K in (1) control the effect of document length and term frequency.

3.2. Segmentation

Since the core task of the SDR track involves the situation where story boundaries are unknown, segmentation of the audio stream assumes some importance. Unlike some other broadcast news speech recognition systems (eg, Odell et al. (1999)), we do not perform any acoustic segmentation in the recognition phase (the audio stream is decoded directly); anyway, there is no good correlation between segments obtained purely from low-level audio features and story segments required for information retrieval. Although other approaches, such as those investigated in the TDT programme, are of some interest, we have no evidence of their suitability for spoken document retrieval.

Thus we have retained the simple approach used last year, based on overlapping rectangular windows of the audio stream[1]. At query time, those relevant segments which overlap are merged. Previously for this type of automatic segmentation, we have used (1) with $b = 0$, since each segment is the same length. However with short segments (30s) this can result in a large number of identical scores, with no good way of breaking the tie. Since the segments do not contain identical numbers of terms — and since we need a tie-breaker — we have used a small non-zero value for b (typically 0.1).

The procedure for merging was as follows. The ranked list of (presumed) relevant segments was processed in best first order. Segments that could be potentially merged with the current segment must: (1) come from the same episode; (2) overlap in time; and (3) be within a rank Δ^r of the current segment. If these conditions are met then the two segments are merged. If the scores of the two segments are within a factor m of each other, and the ranks are within $\Delta^f \leq \Delta^r$ then we assume an *equal* merge; otherwise the higher ranked segment *dominates* the other. In an equal merge, the score of the merged segment is set to be the maximum score of the two segments increased by a factor s, and the reference time is set to be the mid-point of the segment. For a dominating merge, the score and reference time of the merged segment are set to be the same as for the highest scoring component segment. The merging process is iterated until convergence, with parameters Δ^r and Δ^f halved on each iteration.

The overall segmentation procedure is summarized as follows:

1. Entire news episode decoded into a stream of text
2. For indexing, the text stream is split into documents using a fixed length rectangular window with a frame length of t_ℓ and a frame shift of t_s
3. At retrieval time a list of $5R$ segments are retrieved, and the above merging process is carried out. We conducted a number of development experiments to obtain values for the segment merging parameters: $\Delta^r = 1600$, $\Delta^f = 200$, $m = 0.95$ and $s = 1.005$
4. The top R merged segments are then returned.

Previously we have used $t_\ell = 30s$ and $t_s = 15s$. We conducted a variety of experiments looking at the effect of varying the frame shift, with a constant frame length ($t_\ell = 30s$) — our hypothesis was that the possible cost of redundant segments of decreasing the frame shift might be offset by the segment merging algorithm. The results (table 2) indicated a frame shift of $t_s = 9s$ to be a good tradeoff between average precision and index size.

This merging scheme was developed using the TREC-8 development set. Given the several heuristically set parameters, there is a distinct possibility of over-tuning. An alternative approach (Johnson et al., 2000) merged all segments originating within 4 or 5 minutes of each other from a single episode. While this approach may well prove to be robust in actual usage, we believed it may be counter-productive for the SDR track since different relevant documents are sometimes located within less than 4 minutes of each other (owing to adverts, etc.)

---

[1] Also used successfully with an SDR system for a 3 000 hour archive of BBC news broadcasts.

	Short Queries		Terse Queries	
Shift/s	AveP	R-P	AveP	R-P
6	0.526	0.524	0.486	0.490
9	0.526	0.518	0.477	0.485
12	0.518	0.508	0.487	0.476
15	0.510	0.507	0.470	0.477
20	0.498	0.492	0.459	0.467

Table 2: Varying segmentation frame shift (t_s), affect on average precision and R-precision, for development test on TREC-8 queries, with $t_\ell = 30s$.

3.3. Query Expansion

Following experiments on TREC-7 and TREC-8 data (Abberley et al., 1999; Renals et al., 2000) we have applied a query expansion approach whereby the relevance of potential expansion terms to original query terms is obtained by a product of term frequencies weighted by collection frequency weights. Specifically, the query expansion weight $QEW(Q,e)$ for a potential expansion term e and a query Q, across a set of nr (pseudo) relevant documents is defined as:

$$QEW(Q,e) = CFW(e) \sum_{t \in Q} CFW(t) \sum_{i=1}^{nr} TF(e,d_i) \cdot TF(t,d_i). \qquad (4)$$

$QEW(Q,e)$ is used to rank the expansion terms, and the top nt are chosen to expand Q. nr is chosen such that only those documents with a relevance score of greater than $rf \cdot W$ ($rf < 1$) are used. The expanded query terms are weighted by $(nt - rank + 1)/nt$, with terms in the existing query given an additional weight of 1.

In TREC-7 and TREC-8 we obtained significant benefits from query expansion using a parallel corpus largely consisting of newspaper and newswire text from the same period as the target broadcast news corpus. In TREC-9 we constructed a parallel corpus from the following sources:

- TREC-7 SDR reference transcripts (North American broadcast news, covering parts of June 1997 – January 1998): c.0.7M words
- TREC-7 SDR LM text data (LA Times and Washington Post, September 1997 – April 1998): c.14M words.
- TREC-8/9 SDR newswire LM text data (New York Times and AP Newswire, January 1998 – June 1998): c.30M words (AP), c.17M words (NYT).

This gave a total of 135 774 documents with an average document length of 321 words and a standard deviation of 303 words. When carrying out parallel corpus query expansion experiments on TREC-8, we found that 50% of the documents used for QE were from the AP newswire, 36% from the LA Times/Washington Post corpus, 12% from the New York Times and 2% from the TREC-7 reference transcripts.

	Short Queries		Terse Queries	
QE	AveP	R-P	AveP	R-P
None	0.336	0.356	0.351	0.376
Self Only	0.436	0.446	0.432	0.438
Parallel Only	0.499	0.504	0.462	0.476
Self+Parallel	0.490	0.504	0.464	0.478
Self then Parallel	0.490	0.489	0.493	0.492
Parallel then Self	0.526	0.518	0.477	0.485

Table 3: Query expansion using target and parallel corpora, with TREC-8 queries. Self+Parallel indicates that query expansion occurs on a corpus made up of the union of the target and parallel corpora; Self then Parallel indicates that QE is first performed on the target corpus, to produce an expanded query which is then expanded a second time using the parallel corpus. Parallel then Self uses the parallel corpus first, then the target corpus.

In addition to the parallel corpus query expansion, we also experimented with query expansion using blind feedback on the main (target) corpus (also using (4)). Table 3 shows the results of query expansion purely using the target corpus, purely using the parallel corpus and various configurations using both (parallel then self, self then parallel, self and parallel simultaneously). Using our primary metric of average precision with short queries, it appears that expanding the query first on the parallel corpus, then on the target corpus is best. However, this result does not hold for terse queries. So far we have not investigated this effect further. Using a parallel corpus augmented with a copy of the target corpus produced similar results to the parallel corpus alone, as virtually all the documents used for query expansion came from the parallel corpus — probably a side-effect of mixing short 30s segments with whole stories.

4. RESULTS

In the TREC-9 SDR track we performed experiments on the main unknown story boundary (SU) condition and the contrast known story boundary (SK) condition. The same transcriptions were used in each case. Although different text retrieval parameters were used for the SU and SK conditions, the parameters were not dependent on the form of the queries (short or terse). In all cases a query expansion approach of first expanding on the parallel corpus, then on the target corpus was adopted. Table 4 summarizes the parameter settings that we used.

As well as transcriptions produced by our own recogniz-

Parameter	SU	SK
Basic Text Retrieval		
b	0.1	0.7
K	1.0	1.0
Parallel QE		
b^{PQE}	0.7	0.7
K^{PQE}	1.0	1.0
$nrmax^{PQE}$	10	10
nt^{PQE}	20	20
rf^{PQE}	0.75	0.75
Self QE		
b^{SQE}	0.1	0.7
K^{SQE}	1.0	1.0
$nrmax^{SQE}$	40	10
nt^{SQE}	10	10
rf^{PQE}	0.75	0.75
Segment Merging		
Δ^r	1600	–
Δ^f	200	–
m	0.95	–
s	1.005	–

Table 4: Parameters used for TREC-9 Evaluation Runs. b is the length parameter and K the discounting parameter in the weighting function. The additional query expansion parameters are nt (number of terms to add), $nrmax$ (maximum number of pseudo-relevant documents) and rf (multiple of best relevance score that a document must be greater than to be used in QE). The segment merging parameters (described in section 3.2) control the ranking distance threshold below which merging may occur (Δ^r), the ranking difference (Δ^f) and score multiple (m) to determine whether a merge is equal or dominating, and the factor to increase the score by in the case of an equal merge (s).

Transcriptions		Short Queries		Terse Queries	
ID	WER	AveP	R-P	AveP	R-P
R1	10.3	0.409	0.419	0.418	0.425
S1	32.0	0.392	0.399	0.392	0.396
S2	29.2	0.399	0.410	0.393	0.401
B1	26.7	0.387	0.401	0.384	0.398
CUHTK	20.5	0.373	0.388	0.373	0.387
LIMSI1	21.5	0.377	0.405	0.386	0.391
LIMSI2	21.2	0.395	0.407	0.397	0.421

Table 5: TREC-9 SDR track evaluation results for story boundary unknown (SU) condition.

Transcriptions		Short Queries		Terse Queries	
ID	WER	AveP	R-P	AveP	R-P
R1	10.3	0.509	0.489	0.492	0.477
S1	32.0	0.464	0.441	0.475	0.463
S2	29.2	0.465	0.435	0.478	0.463
B1	26.7	0.462	0.447	0.469	0.451

Table 6: TREC-9 SDR track evaluation results for story boundary known (SK) condition.

ers (S1 and S2), we also used the following transcriptions:

1. Reference transcriptions prepared from closed captions (R1);
2. Baseline speech recognizer transcription prepared by NIST (B1);
3. Speech recognition transcriptions prepared by LIMSI (LIMSI1 and LIMSI2);
4. Speech recognition transcriptions prepared by Cambridge University (CUHTK).

Results for the SU case in the TREC-9 SDR track are presented in table 5. The average precisions are, in all cases, 20-25% lower (relative) than for TREC-8. This suggests that the TREC-9 queries may have been more difficult in some way, or that the system was over-tuned to the TREC-8 queries. Secondly, we see that the performance on the terse queries is similar to that on short queries. Note that we optimised our system using short queries on TREC-8.

Finally, following the trend of previous evaluations, the link between word error rate and text retrieval accuracy is very weak. Indeed, out of all the speech recognition transcriptions, the highest average precision on short queries is achieved using S2 (with a WER of 29%).

For contrast, results for the SK case in the TREC-9 SDR track are presented in table 6. These results follow the same form as the SU results, indicating that the low average precisions (compared with TREC-8) are not due to the segmentation/merging procedure. The relative gap between SK and SU average precision is 10–20%.

5. DISCUSSION

5.1. Terse and Short Queries

The topics for which we get a substantially better performance from short queries compared with terse queries fall into two basic types: those where the terse query is little more than an abbreviation — eg, "I L O" (130), "N B A" (165) – and those where the terse query is expressed using different words or incompletely compared with the short query (136,155,156,171). The first case might be improved by better acronym processing, the second ought to be dealt with by query expansion. More analysis is required.

5.2. Query Expansion

Previously we used parallel corpus QE only, having found that query expansion on the target corpus did not give a reliable improvement. From our experiments on TREC-8, it seems that first expanding on the parallel corpus, with the resultant expanded query being expanded again using the target corpus gives a reliable improvement. An interesting factor to be investigated is that query expansion seems to have different behaviour on terse and short queries.

5.3. Non-lexical Information

Although we were able to compute various types of non-lexical information (eg, named entities, speaker changes, sentence boundaries) we chose not to use such information in this evaluation. In the case of named entities, this was because we did not have a principled way of using them. In the case of richer boundary information, we did not feel that this would be rewarded under the evaluation metrics in use. For example, in discussions with broadcast archive users of our system, it has been apparent that returning clips that begin and end at natural boundaries would enhance their appreciation of the system; the single reference time method of denoting segments does not give any credit for accurate begin/end points.

5.4. Standard QE Corpus

A great deal of effort has gone into standardizing the acoustic model and LM training data for speech recognition, to enable better evaluation of the underlying models and algorithms. It would be of interest to increase this standardization, by specifying a baseline query expansion corpus, to be used in a contrast run (at least).

6. CONCLUSION

Our major conclusions are as follows:

- There is only a weak link between speech recognition accuracy and spoken document retrieval precision and recall;
- Query expansion using both a parallel text corpus and the target corpus is reliable and extremely effective;
- Simple fixed segmentation, followed by query-time segment merging is reliable, causing a degradation of 10–20% compared with the hand-segmented case.

ACKNOWLEDGEMENTS

Dan Ellis and Tony Robinson worked on the system for TREC-8 SDR and the work described here uses the fruits of their labour. We thank Tony Robinson and SoftSound for use of the CHRONOS decoder. This work was supported by ESPRIT project THISL (EP23495) and EPSRC project SToBS (GR/M36717).

REFERENCES

Abberley, D., S. Renals, G. Cook, and T. Robinson (1999). Retrieval of broadcast news documents with the THISL system. In *Proc. Seventh Text Retrieval Conference (TREC-7)*, pp. 181–190.

Bourlard, H. and N. Morgan (1994). *Connectionist Speech Recognition—A Hybrid Approach*. Kluwer Academic.

Clarkson, P. and R. Rosenfeld (1997). Statistical language modeling using the CMU-Cambridge toolkit. In *Proc. Eurospeech*, pp. 2707–2710.

Cook, G., K. Al-Ghoneim, D. Ellis, E. Fosler-Lussier, Y. Gotoh, B. Kingsbury, N. Morgan, S. Renals, T. Robinson, and G. Williams (1999). The SPRACH system for the transcription of broadcast news. In *Proc. DARPA Broadcast News Workshop*, pp. 161–166.

Hennebert, J., C. Ris, H. Bourlard, S. Renals, and N. Morgan (1997). Estimation of global posteriors and forward-backward training of hybrid HMM/ANN systems. In *Proc. Eurospeech*, Rhodes, pp. 1951–1954.

Hermansky, H. (1990). Perceptual linear predictive (PLP) analysis of speech. *J. Acoust. Soc. Amer. 87*, 1738–1752.

Johnson, S. E., P. Jourlin, G. L. Moore, K. Spärck Jones, and P. C. Woodland (2000). Audio indexing and retrieval of complete broadcast news shows. In *Proc. RIAO 2000, Content Based Mulitmedia Information Access*, pp. 1163–1177.

Kingsbury, B. E. D., N. Morgan, and S. Greenberg (1998). Robust speech recognition using the modulation spectrogram. *Speech Communication 25*, 117–132.

Odell, J. J., P. C. Woodland, and T. Hain (1999). The CUHTK-Entropic 10xRT broadcast news transcription system. In *Proc. DARPA Broadcast News Workshop*, pp. 271–275.

Renals, S., D. Abberley, D. Kirby, and T. Robinson (2000). Indexing and retrieval of broadcast news. *Speech Communication 32*, 5–20.

Renals, S. and M. Hochberg (1999). Start-synchronous search for large vocabulary continuous speech recognition. *IEEE Trans. Speech and Audio Processing* 7, 542–553.

Robertson, S. E. and K. Spärck Jones (1997). Simple proven approaches to text retrieval. Technical Report TR356, Cambridge University Computer Laboratory.

Robinson, A. J. (1994). The application of recurrent nets to phone probability estimation. *IEEE Trans. on Neural Networks* 5, 298–305.

Robinson, A. J., G. D. Cook, D. P. W. Ellis, E. Fosler-Lussier, S. J. Renals, and D. A. G. Williams (2001). Connectionist speech recognition of broadcast news. *Speech Communication*. In press.

Robinson, T. and J. Christie (1998). Time-first search for large vocabulary speech recognition. In *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 829–832.

Robinson, T., M. Hochberg, and S. Renals (1996). The use of recurrent networks in continuous speech recognition. In C.-H. Lee, K. K. Paliwal, and F. K. Soong (Eds.), *Automatic Speech and Speaker Recognition – Advanced Topics*, pp. 233–258. Kluwer Academic Publishers.

Sheffield Interactive Experiment at TREC-9

M. Beaulieu, H. Fowkes and H. Joho
Department of Information Studies
University of Sheffield, U.K.
m.beaulieu@sheffield.ac.uk

Abstract

The paper reports on the experiment conducted by the University of Sheffield in the Interactive Track of TREC-9 based on the Okapi probabilistic ranking system. A failure analysis of results was undertaken to correlate search outcomes with query characteristics. A detailed comparison of Sheffield results with the aggregate for the track reveals that the time element, topic type, and searcher characteristics and behaviour are interdependent success factors. An analysis of the ranking of documents retrieved by the Okapi system and deemed relevant by the assessors also revealed that more than 50% appeared in the top 10 and 80% in the top 30. However the searchers did not necessarily view these and over half of the items deemed relevant by the assessors and examined by the searchers were actually rejected.

1. Introduction

The experiment for TREC-9 as in previous rounds in which Sheffield has participated was based on the Okapi system. Although the experimental design included two versions of the system, one with relevance feedback and one without, it was envisaged that the five minute time limit for searching each of the interactive queries for Trec-9 would offer little opportunity for searchers to use the feedback facility for query reformulation. Our aim was thus to focus on the characteristics of the two types of queries introduced for the TREC-9 interactive task and assess their relative impact on the performance of both the searchers and the system.

The graphical user interface of the Okapi systems remained exactly the same as in the last three rounds of TREC and includes:

- a query box
- a working query window containing system generated candidate terms for query expansion
- a scrollable window displaying a ranked list of the top fifty retrieved items
- a window for collecting documents marked as relevant and saved by the searcher
- a separate overlapping window for viewing items selected from the hitlist where searchers have to make a relevance judgement.

The standard questionnaires for the interactive track were used for data collection including: session entry, pre-search, post search, post-system and session exit. In addition transaction logs and talk aloud protocols provided system data and user perceptions in the course of the search. However subjects were not very forthcoming in talking aloud due to the time constraint imposed on them and consequently the protocols provided very limited insight.

Sixteen searchers participated in the experiment and fourteen were Masters students in the Information Studies Department. None had used the Okapi system before, although most had some knowledge of ranking systems either through their general use of search engines on the Web or through a course on information retrieval in their programme of study. Half had between two to three year's online experience, with searching the Web and library catalogues being the most common types of systems. The other half was deemed to be novice users with a year or less experience.

2. Results and query characteristics

The TREC-9 interactive task included two types of topics. For the first set 901-904, searchers had to find a given number of different answers to a question, e.g. *three* national parks, *a* Roman site in France, *four* Orson Welles films, and *three* countries importing Cuban sugar. In essence these topics were not dissimilar from those used in the Interactive Track for TREC-7 and TREC-8, where searchers had to find as many different instances or answers as possible. The main difference in TREC-9 was that searchers had only five minutes to complete the task as opposed to twenty minutes in the previous rounds.

The second set of queries 905-908 required a single correct answer between two possible choices, e.g. the *longest* running TV programme, the painting completed *first*, the *last* Chinese dynasty, the country with the *larger* population. In arriving at a correct answer searchers had to find appropriate supporting evidence in different documents and save those documents.

The results of the Sheffield respondents compared to the aggregate performance of the participants in the track are presented in Tables 1a, 1b. The following will discuss these results in relation to the characteristics of each of the eight individual topics.

Table 1a: Sheffield results compared to the aggregate for Type 1 topics, 901-904

Response / Topic number	**901**		**902**		**903**		**904**	
	Shef	Agg	Shef	Agg	Shef	Agg	Shef	Agg
All answers are supplied and supported *(2,2)*	-	8 *(7%)*	4 *(25%)*	18 *(18%)*	-	3 *(3%)*	1 *(6%)*	29 *(27%)*
All answers are supplied and some supported *(2,1)*	-	-	-	-	-	2 *(2%)*	*1* *(6%)*	7 *(7%)*
All answers are supplied and none are supported *(2,0)*	-	3 *(3%)*	-	-	-	1 *(1%)*		

Some answers are supplied, and all are supported *(1,2)*	1 *(6%)*	38 *(35%)*	-	-	8 *(50%)*	44 *(41%)*	7 *(44%)*	35 *(33%)*
Some answers are supplied and some supported *(1,1)*	1 *(6%)*	2 *(2%)*	-	-	3 *(19%)*	23 *(22%)*	2 *(13%)*	14 *(13%)*
Some answers are supplied and none are supported *(1,0)*	1 *(6%)*	6 *(6%)*	-	-	3 *(19%)*	27 *(25%)*	1 *(6%)*	8 *(8%)*
No answers are supplied and none are supported *(0,0)*	13 *(82%)*	50 *(47%)*	12 *(75%)*	80 *(82%)*	2 *(12%)*	6 *(6%)*	4 *(25%)*	13 *(12%)*
Total number of searchers	16	107	16	98	16	106	16	106

Table 1b: Sheffield results compared to the aggregate for type 2 topics 905-908.

Response / Topic number	**905**		**906**		**907**		**908**	
	Shef	**Agg**	**Shef**	**Agg**	**Shef**	**Agg**	**Shef**	**Agg**
All answers are supplied and supported *(2,2)*	8 *(50%)*	65 *(61%)*	8 *(50%)*	41 *(41%)*	9 *(56%)*	77 *(74%)*	-	9 *(25%)*
All answers are supplied and none are supported *(2,0)*	3 *(19%)*	9 *(9%)*	4 *(25%)*	32 *(22%)*	7 *(44%)*	15 *(14%)*	5 *(31%)*	-
No answers are supplied and none are supported *(0,0)*	5 *(31%)*	32 *(30%)*	4 *(25%)*	37 *(37%)*	-	13 *(12%)*	11 *(69%)*	78 *(75%)*
Total number of searchers	16	106	16	100	16	105	16	101

901: What are the names of the three US national parks where one can find redwoods?

Sheffield respondents performed poorly on this query compared to the aggregate with 13 out of 16 finding no correct answers and the remaining 3 providing only partial answers, one of which was unsupported. The nil answers, which were twice as high as the aggregate, appear to have been influenced by some ambiguity in differentiating the meaning between "national" and "state" parks. The question may have presented some cultural bias, as a high proportion of our searchers were international students with no previous knowledge of the topic as indicated in the pre-search questionnaire.

902: Identify a site with Roman ruins in present day France?

The Sheffield results are comparable to the rest of the track with a quarter successful answers and three quarters of searchers unable to find a correct answer. The polarised results may be due to the combination of evidence required to arrive at an answer, i.e. the name of the country, the specific location as well as the type of ruin. Furthermore only 7 documents were identified by the TREC assessors as providing an answer in the retrieved pool, a small number compared to other topics (See Table 3).

903: Name four films in which Orson Welles appeared

Once again our searchers produced comparable results with 13 out of 16 producing partial answers, but only half provided partial supporting evidence. This question was somewhat of a trick question in that most references referred to films directed by Welles and it would appear that there was some amount of guesswork in identifying films in which he was also an actor. The one Sheffield searcher, who got all the correct answers with supporting documents, had a special interest in film studies and was confident about the answer prior to searching the system. Three other searchers had also indicated pre-knowledge on this topic with a high degree of confidence.

904: Name three countries that imported Cuban sugar during the period of time covered by the document collection

Only one Sheffield searcher identified three countries compared with over a quarter of the aggregate. In addition twice as many Sheffield searchers did not succeed in finding any answers at all, 25% compared to 12%. This topic was also undertaken in TREC-8. The performance then was equally poor even though searchers had twenty minutes to search. It was found that although searchers were essentially looking for labels, i.e. names of countries, they had to engage with the content of the document to ensure that the correct context was covered. Although the time limit may have been a factor in TREC-9, it obviously doesn't account for the poorer performance compared to other participants in the track.

905: Which children's TV program was on air longer: the original Mickey Mouse Club or the original Howdy/Doody show?

Comparable results were obtained with the overall track with half of the searchers choosing the correct answer with supporting evidence. However a third of all searchers in the track provided no answer at all. As in question 902 on Roman ruins in France, few relevant documents provided the answer (See Table 3). In fact the searchers commonly saved two documents, one was deemed by the assessors to support the answer whereas the other didn't.

906: Which painting did Edward Munch complete first: Vampire or Puberty?

Sheffield performed slightly better than the aggregate with 50% getting the right answer with the correct supporting documents and 25% not finding the answer. Surprisingly 25% provided the right answer with no correct supporting evidence. Since only three documents were judged to be relevant by the assessors (see Table 3), it would appear that searchers were able to make correct deductions or an informed guess.

907: Which was the dynasty of China: Qing or Ming?

Sheffield searchers outperformed the aggregate on this query with all identifying the correct answer, although just under half did not back it up with correct documents. Five of our searchers indicated that they knew the answer before searching, which may in part account for this discrepancy.

908: Is Denmark larger or smaller in population than Norway?

Just under a third of Sheffield searchers got the right answer but without supporting evidence compared with a quarter in the overall track who did provide correct supporting evidence. However in both cases around three quarters failed to find the answer all together. Again the high failure rate could have been related to the need to piece together different evidence over multiple documents in a short space of time.

Table 2 presents a summary of the adjusted score obtained for each answer which was correctly identified and supported by an appropriate document. The difference in the level of performance between the two different types of topics 901- 904 and 905-908 are clearly demarcated and reflect the overall pattern of performance in the track. It may be that the time limit was a critical success factor whereby it was more difficult to find correct multiple answers in the first type of topic and easier to find single answers in the second type.

Table 2: Sheffield adjusted score for correct supported answers for each topic out of the maximum obtainable score.

Topic no	**901**	**902**	**903**	**904**	**905**	**906**	**907**	**908**
Adjusted score	3 out of *48* *(6%)*	4 out of *16* *(25%)*	19 out of *64* *(30%)*	23 out of *48* *(48%)*	8 out of *16* *(50%)*	8 out of *16* *(50%)*	9 out of *16* *(56%)*	0 out of *16* *(0%)*

3. Searcher performance vs system performance

In an attempt to isolate user effect on system performance, the session logs were analysed to ascertain what proportion of relevant documents identified by the assessors were actually retrieved by the system. Table 3 compares the number of documents judged as relevant by the assessors for each topic and the average retrieved by the system in the initial ranked hitlist of the top 50 documents retrieved for all of the searches. It would appear that poor searcher performance reported in Table 2 for topics 901 and 908 are not really borne out it terms of the average number of actual relevant documents retrieved by the system in the retrieved sets of 50 documents displayed to the searcher. Three quarters or more unique assessed relevant documents are retrieved by the system in all but one topic (908) in the first iteration which provides some evidence of the system's high level of performance.

Table 3 Assessed relevant documents retrieved in the top 50.

Topic no	Total number of unique assessed relevant docs out of the possible maximum	Average no of assessed relevant docs retrieved
901	10/13 *(77%)*	5.6 *(43%)*
902	6/ 7 *(86%)*	1.87 (27%)
903	13/17 *(76%)*	5.6 *(33%)*
904	29/39 (74%)	11 *(44%)*
905	7/7 *(100%)*	3.2 *(46%)*
906	3/3 *(100%)*	2.4 *(80%)*
907	20/23 *(87%)*	7 *(30%)*
908	9/15 *(60%)*	2.6 (17%)

Tables 4a, 4b compare the total number of assessed relevant documents examined by searchers for each of the topics with the number actually saved or deemed relevant by the searchers and those which were not deemed to be relevant. Overall 53% of documents deemed relevant by the assessors were examined but actually rejected by searchers. There were more documents rejected for type 1 topics than for type 2, 46% compared to 39%.

Table *4a:* Assessed relevant documents viewed and saved in the top 50, Type 1 Topics 901-904

Topic no	**No of relevant docs viewed**	**No of relevant docs saved**	**No of relevant docs not saved**
901	29	13 *(45%)*	16 *(55%)*
902	12	4 *(33%)*	8 *(67%)*
903	15	11 *(73%)*	4 *(27%)*
904	24	10 *(42%)*	14 *(58%)*

Total	**70**	**38** ***(54%)***	**42** ***(46%)***

Table *4b*: Assessed relevant documents viewed and saved in the top 50, Type 2 Topics 905-908

Topic no	**No of documents viewed**	**No of documents saved**	**No of documents Not saved**
905	25	14 *(56%)*	11 *(44%)*
906	18	9 *(50%)*	9 *(50%)*
907	9	8 *(89%)*	1 *(11%)*
908	2	2 *(100%)*	-
Total	**54**	**33** ***(61%)***	**21 (39%)**
Overall Total	**134**	**71** ***(47%)***	**63** ***(53%)***

Table 5 presents the ranking position of all the assessed relevant documents retrieved by the system but not necessarily viewed by the searchers. More than half appeared in the top 10 of the hitlist displayed to the searchers and 80% in the top 30.

Table 5: Assessed relevant document ranking for all searches for each topic.

Topic	**Top 10**	**Top 20**	**Top 30**	**Top 40**	**Top 50**
901	55 *(65%)*	8 *(9%)*	3 *(4%)*	1 *(1%)*	18 *(21%)*
902	19 *(76%)*	4 *(16%)*	2 *(8%)*	-	-
903	37*(44%)*	15 *(18%)*	13 *(15%)*	10 *(12%)*	9 *(11%)*
904	53 *(36%)*	30 *(20%)*	29 *(20%)*	17 *11(%)*	20 *(13%)*
905	38 *(70%)*	7 *(13%)*	5 *(9%)*	3 *(6%)*	1 *(2%)*
906	31 (97%)	1(3%)	-	-	-
907	36 *(43%)*	14 *(17%)*	13 *(15%)*	10 *(12%)*	11 *(13%)*
908	19 *(59%)*	-	7 (22%)	3 *(9.5%)*	3 *(9.5%)*
Totals	**288** ***(53%)***	**79** ***(14%)***	**72** ***(13%)***	**44** ***(9%)***	**62** ***(11%)***

4. Discussion of success factors

Although more failure analysis can be carried out on the data, a number of interdependent success factors appear to contribute to the above results including: time, topic type, and searcher characteristics and behaviour. Firstly there appeared to be some degree of correlation between topic type and the amount of time available for searching. 28% of searchers indicated that they didn't have enough time to undertake type 1 topics and 14% deemed they had enough time. In the case of type 2 topics there was little difference between those who felt they had enough or not enough time, 20% as opposed to18%. Searchers' perceptions regarding the time available also related to their level of satisfaction with the search outcome. There appeared to be a higher degree of confidence in the outcome of type 2 topics.

The discrepancy between the overall track performance in the two types of topics is not easily reconciled with our findings in the analysis of the degree of complexity of the TREC-8 topics and searching behaviour (1). In TREC-8 we found that in order to arrive at a relevance judgement, more complex topics required some interpretation on the part of the searcher and a higher degree of engagement with the contents of documents being examined. Less complex topics on the other hand were more easily understood at the outset and by enlarge relevant documents were identified by scanning for highlighted query terms in the documents. Hence it could be said that in general complex topics are likely to require more effort from the searcher than less complex ones. In the current round the differences in the level of engagement with the documents was not easily discernible in the time allowed for each search. However it would seem that the number of different answers required for type 1 topics was more demanding than the single answer required for type 2. The short time element may have been a more important success factor here than the complexity of the topic.

A third element, which contributed to the success/failure of the search outcomes, relates to the behaviour and the characteristics of the searchers themselves. Although type 2 topics required searchers to engage with the documents viewed to accumulate evidence for the correct answer, there was a substantial number of correct answers from Sheffield searchers which were not supported by appropriate documents 30% (Table 1b). The reason could be two-fold: firstly informed guesses could be made on partial evidence and secondly it may have been difficult for searchers to ascertain which document provided the correct evidence. The number of assessed relevant documents, which were viewed and rejected by searchers, would support this element of uncertainty, i.e. the difficulty in identifying the correct evidence and knowing which documents to save. In comparing the system performance with regard to actual assessed relevant documents retrieved by the system and the actual search outcomes for the topics, it is clear that search outcomes are highly dependent on the searchers themselves. Searchers either fail to examine relevant documents, or disagree with the assessors' judgements.

5. Conclusions

Since the Interactive Track was first established, much effort has been put into defining an appropriate search task. Although there is evidence to show that a realistic and reliable experimental setting can be created through simulated tasks (2), the search task for the current round of the Interactive Track was not ideal. In particular whilst it may be a common and realistic scenario for a searcher to want to find an answer as quickly as possible, the five minute time constraint in an experimental setting had an adverse effect. The participants in the experiment not only had to find the correct answer(s) but also had to provide the correct evidence, i.e. identify the documents which provided the right answer. Providing the evidence proved to be difficult and led to second guessing. With hindsight it may have been better for searchers to have had more time to engage with the documents to avoid readily rejecting items which did in fact contain the supporting evidence.

In addition to highlighting the limitations of the task, the current experiment also demonstrated the importance of comparing both user and systems performance in interactive searching. Although it is recognised a ranked output may not be the best way of presenting results (3), little research has been carried out to date on how searchers handle and interpret ranked output.

References

1. Fowkes, H. & Beaulieu, M. Interactive searching behaviour: Okapi experiment for TREC-8. In: *Proceedings of the BCS-IRSG 22nd Annual Colloquium on Information Retrieval Research*, Cambridge, UK, 5th-7th April, 2000. 47-56.

2. Borlund, P. & Ingwersen, P. (1997) The Development of a Method for the Evaluation of Interactive Retrieval Systems In: *Journal of Documentation* 53 (3) 225-250.

3. Hearst, M. (1999) User Interfaces and Visualisation, In: *Modern Information Retrieval* (eds) Baeza-Yates, R. & Ribeiro-Neto, B., Addison-Wesley Longman Publishing Company.

Question Answering in Webclopedia

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Michael Junk, Chin-Yew Lin

Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292-6695
Tel: 310-448-8731
Fax: 310-823-6714
Email: {hovy,gerber,ulf,junk,cyl}@isi.edu

1. Introduction: Question Answering

IR techniques have proven quite successful at locating within large collections of documents those relevant to a user's query. Often, however, the user wants not whole documents but brief answers to specific questions: *How old is the President? Who was the second person on the moon? When was the storming of the Bastille?* Recently, a number of research projects have investigated the computational techniques needed for effective performance at this level of granularity, focusing just on questions that can be answered in a few words taken as a passage directly from a single text (leaving aside, for the moment, the answering of longer, more complex answers, such as stories about events, descriptions of objects, compare&contrast discussions, arguments of opinion, etc.).

The systems being built in these projects exhibit a fairly standard structure: all create a query from the user's question, perform IR with the query to locate (segments of) documents likely to contain an answer, and then pinpoint the most likely answer passage within the candidate documents. The most common difference of approach lies in the pinpointing. A 'pure IR' approach would segment each document in the collection into a series of mini-documents, retrieve the segments that best match the query, and return them as answer. The challenge here would be to make segments so small as to be just answer-sized but still large enough to be indexable. A 'pure NLP' approach would be to match the parse and/or semantic interpretation of the question against the parse and/or semantic interpretation of each sentence in the candidate answer-containing documents, and return the best match(es). The challenge here would be to perform parsing, interpretation, and matching fast enough to be practical, given the large volumes of text to be handled.

Answering short questions thus becomes a problem of finding the best combination of word-level (IR) and syntactic/semantic-level (NLP) techniques, the former to produce as short a set of likely candidate segments as possible and the latter to pinpoint the answer(s) as accurately as possible.

Because language allows paraphrasing and inference, however, working out the details is not entirely straightforward. In this paper we describe the Webclopedia, a system that uses a classification of QA types to facilitate coverage, uses a robust syntactic-semantic parser to perform the analysis, and contains a matcher that combines word- and parse-tree-level information to identify answer passages. Section 2 outlines the Webclopedia approach and architecture; Section 3 describes document retrieval and processing, Section 4 describes the QA Typology, Section 5 the parsing, and Section 6 the matching.

2. Webclopedia

Webclopedia's architecture, shown in Figure 1, follows the pattern outlined in Section 1:

Parsing of question: The CONTEX parser (see Section 5) is used to parse and analyze the question, assisted by BBN's IdentiFinder (Bikel et al., 1999).

Question analysis: To form a query, single- and multi-word units (content words) are extracted from the parsed query. WordNet synsets are used for query expansion. See Section 3.

IR: The IR engine MG (Witten et al. 1994) is used to return and rank the top 1000 documents.

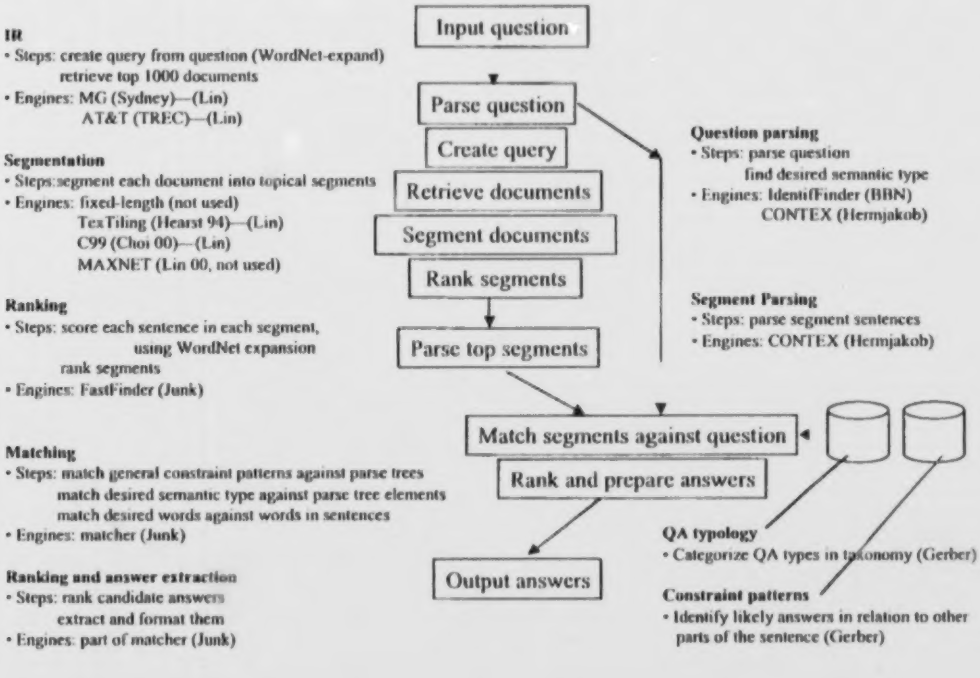
Segmentation: To decrease the amount of text to be processed, the documents are broken into semantically coherent segments. Two text segmenters were tried—TexTiling (Hearst, 94), C99 (Choi, 00); the first is used.

Ranking of segments: For each segment, each sentence is scored using a formula that rewards word and phrase overlap with the question and expanded query words. The segments are ranked.

Parsing of segments: CONTEX also parses each sentence of the top-ranked 100 segments.

Pinpointing: For each sentence, three steps of matching are performed (see Section 6); two compare the parses of the question and the sentence; the third moves a fixed-length window over each sentence and computes a goodness score based on the words and phrases contained in it.

Ranking of answers: The candidate answers' scores are compared and the winning answer(s) are output.



IR
• Steps: create query from question (WordNet-expand)
retrieve top 1000 documents
• Engines: MG (Sydney)—(Lin)
AT&T (TREC)—(Lin)
Segmentation
• Steps:segment each document into topical segments
• Engines: fixed-length (not used)
TexTiling (Hearst 94)—(Lin)
C99 (Choi 00)—(Lin)
MAXNET (Lin 00, not used)
Ranking
• Steps: score each sentence in each segment,
using WordNet expansion
rank segments
• Engines: FastFinder (Junk)
Matching
• Steps: match general constraint patterns against parse trees
match desired semantic type against parse tree elements
match desired words against words in sentences
• Engines: matcher (Junk)
Ranking and answer extraction
• Steps: rank candidate answers
extract and format them
• Engines: part of matcher (Junk)
Input question
Parse question
Create query
Retrieve documents
Segment documents
Rank segments
Parse top segments
Match segments against question
Rank and prepare answers
Output answers
Question parsing
• Steps: parse question
find desired semantic type
• Engines: IdentifFinder (BBN)
CONTEX (Hermjakob)
Segment Parsing
• Steps: parse segment sentences
• Engines: CONTEX (Hermjakob)
QA typology
• Categorize QA types in taxonomy (Gerber)
Constraint patterns
• Identify likely answers in relation to other parts of the sentence (Gerber)

Figure 1. Webclopedia architecture.

3. Information Retrieval and Document Ranking

Analyzing the Question to Create a Query

We parse input questions using CONTEX (Section 5) to obtain a semantic representation of the questions. For example, we determine that the question "Who is Johnny Mathis' high school track coach?" is asking for the name of person. The question analysis module identifies noun phrases, nouns, verb phrases, verbs, adjective phrases, and adjectives embedded in the question. These phrases/words are assigned significance scores according to the frequency of their type in our question corpus (a collection of 27,000+ questions and answers), secondarily by their length, and finally by their significance scores, derived from word frequencies in the question corpus.

We remain indebted to BBN for the use of IdentiFinder (Bikel et al., 1999), which isolates proper names in a text and classifies them as person, organization, or location.

Expanding Queries

In order to boost recall we use WordNet 1.6 (Fellbaum 1998) to expand query terms and place all the expanded terms into a Boolean expression. For example, "high school" is expanded to:

"(high&school)|(senior&high&school)|(senior&high)|high|highschool"

It is obvious that such brute force expansion has undesirable effects. The expanded "high school" query contains "high". This will make "high school" relatively less significant, since "high" is a very common word. We did not try to fix this problem in this year's TREC evaluation, but are planning to improve the expansion procedure next year.

Retrieving Documents

We use MG (Witten et al. 1994) as our search engine. Although MG is capable of performing ranked query, we only use its Boolean query capability. For the entire TREC9 test corpus, the size of the inverse index file is about 200 MB and the size of the compressed text database is about 884 MB. The stemming option is turned on. Queries are sent to the MG database, and the retrieved documents are ranked according to their ranking from query analysis. For example:

Johnny&mathis&((high&school)|(senior&high&school)|(senior&high)|high|highschool)

will be sent to the database first. If the number of documents returned is less than a pre-specified threshold then we retain this set of documents as the basis for further processing. The threshold is set to 5,000 in our TREC9 evaluation. If nothing is returned then we relax the query by taking the next query term in our query rank list. In this case, it is "high school track coach". If more than 5,000 documents are returned we drop the query expansion and use the original query terms instead. For this example, the query will be "Johnny&mathis&high&school&track&coach".

In some cases, it is impossible to get the number of returned documents down to 5,000. For example, the question "What is the meaning of life?" will return an enormous amount of documents since all the words in the query are very common. We plan to address this problem by adding proximity and order constraints to the query process.

Ranking Documents

If the total numbers of documents returned by MG is N, we would like to rank the documents to maximize answer recall and precision in the topmost K << N, in order to minimize the parsing and subsequent processing. In this phase we set K=1,000. Our document ranker uses the following scoring method:

- Each question word gets a score of 2
- Each synonym gets a score of 1
- Other words get a score of 0

Normally common words are ignored unless they are part of a phrase in question word order, in which case they get a score of 2 along with other words in the phrase. Based on these scores, the total score for a document is:

Document score = sum of word scores / number of different words

Segmenting Documents

Splitting each document into topical segments to be input to the matcher is based on the assumption that important contextual information for pinpointing answers tends to occur within a local context. This is mostly true for the setup of TREC9 Q&A. Furthermore, CONTEX does not use information outside sentence boundaries. This step helps the system focus on smaller regions of text where answers are most likely to be found.

We tried two text segmenters, TextTiling (Hearst 1994) and C99 (Choi 2000). They perform at almost the same level, though TextTiling is faster.

Ranking Segments

The resulting segments are re-ranked using the same ranker described earlier. This time, only the topmost 100 segments are passed to the parser (and then to the matcher for answer pinpointing).

Retrieval Results

We evaluated our IR front end in 6 separate experiments using the 238 training questions obtained from NIST. The resulting answer distributions within the top 1,000 segments are shown in Table 1.

N <=	5	10	20	30	40	50	60	70	80	90	100	500	1000
Test0=38	12	15	22	23	25	27	28	28	29	29	30	36	36
%	19%	23%	34%	36%	39%	42%	44%	44%	45%	45%	47%	56%	56%
Test1=52	27	31	38	39	41	41	41	41	41	41	41	48	48
%	42%	48%	59%	61%	64%	64%	64%	64%	64%	64%	64%	75%	75%
Test2=64	23	29	38	41	43	46	47	47	47	48	48	56	56
%	36%	45%	59%	64%	67%	72%	73%	73%	73%	75%	75%	88%	88%
Test3=52	17	21	24	27	29	30	31	32	32	32	32	44	46
%	33%	40%	46%	52%	56%	58%	60%	62%	62%	62%	62%	85%	89%
Test4=55	34	39	46	48	48	50	50	50	50	51	51	54	54
%	62%	71%	84%	87%	87%	91%	91%	91%	91%	93%	93%	98%	98%
Test5=54	25	30	34	38	40	41	42	43	43	45	45	51	51
%	46%	56%	63%	70%	74%	76%	78%	80%	80%	83%	83%	94%	94%
Overall	138	165	202	216	226	235	239	241	242	246	247	289	291
%	44%	52%	64%	69%	72%	75%	76%	77%	77%	78%	78%	92%	92%

Table 1. Percentage of topmost N segments containing an answer after retrieval and ranking.

It is interesting to see that the system gets about 52% of answer segments within the top 10 and reaches only 78% within the top 100. And even in the top 1000 segments, 8% of the answers are missing. This indicates that further improvement of the IR front end is critical.

4. The QA Typology

There are many ways to ask the same thing. Likewise, there are many ways of delivering the same answer. Such variations form a sort of semantic equivalence class of both questions and answers; speaking approximately, any form of the question can be answered by any form of the answer. Since the user may employ any version of his or her question, and the source documents may contain any version(s) of the answer, an efficient system should group together equivalent question types and answer types. Any specific question can then be indexed into its type, from which all equivalent forms of the answer can be ascertained. These QA equivalence types can help with both query expansion (for IR) and answer pinpointing (for NLP).

However, the equivalence is fuzzy; even slight variations introduce exceptions: *who invented the gas laser?* can be answered by both *Ali Javan* and *a scientist at MIT*, while *what is the name of the person who invented the gas laser?* requires the former only. This inexactness suggests that the QA types be organized in an inheritance hierarchy, allowing the answer requirements satisfying more general questions to be overridden by more specific ones 'lower down'.

Previous work in automated question answering has often categorized questions by question word alone or by a mixture of question word and the semantic class of the answer (Srihari and Li, 2000; Moldovan et al., 2000). To ensure full coverage of all forms of simple question and answer, we have been developing a QA Typology as a taxonomy of QA types, becoming increasingly specific as one moves from root downward. Instead of focusing on question word or semantic type of the answer, our classes attempt to represent the user's intention, including for example the classes Why-Famous (for *Who was Christopher Columbus?* but not *Who discovered America?*, which is a Proper-Person QA type) and Abbreviation-Expansion (for *What does NAACL stand for?*).

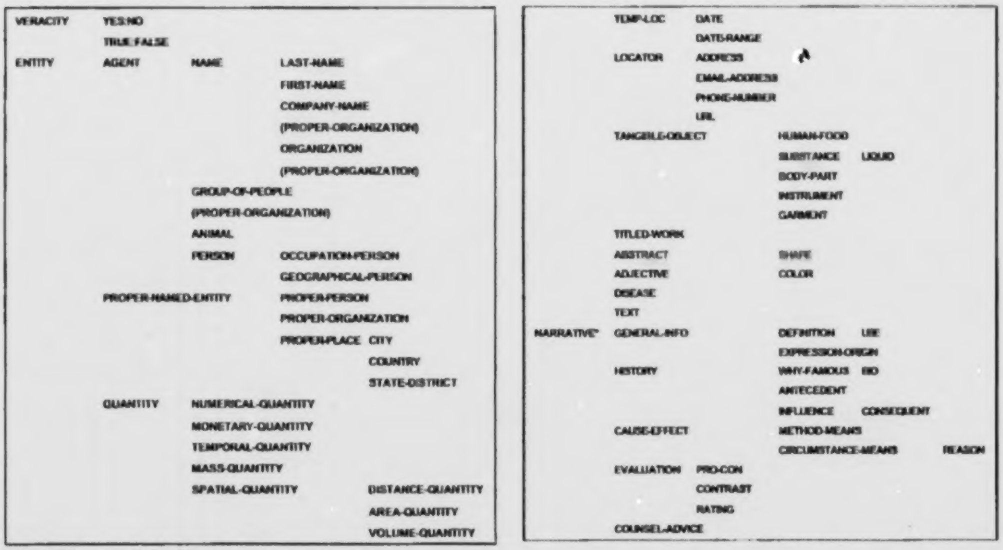

VERACITY
YES:NO
TRUE:FALSE
ENTITY
AGENT
NAME
LAST-NAME
FIRST-NAME
COMPANY-NAME
(PROPER-ORGANIZATION)
ORGANIZATION
(PROPER-ORGANIZATION)
GROUP-OF-PEOPLE
(PROPER-ORGANIZATION)
ANIMAL
PERSON
OCCUPATION-PERSON
GEOGRAPHICAL-PERSON
PROPER-NAMED-ENTITY
PROPER-PERSON
PROPER-ORGANIZATION
PROPER-PLACE
CITY
COUNTRY
STATE-DISTRICT
QUANTITY
NUMERICAL-QUANTITY
MONETARY-QUANTITY
TEMPORAL-QUANTITY
MASS-QUANTITY
SPATIAL-QUANTITY
DISTANCE-QUANTITY
AREA-QUANTITY
VOLUME-QUANTITY
TEMP-LOC
DATE
DATE-RANGE
LOCATOR
ADDRESS
EMAIL-ADDRESS
PHONE-NUMBER
URL
TANGIBLE-OBJECT
HUMAN-FOOD
SUBSTANCE
LIQUID
BODY-PART
INSTRUMENT
GARMENT
TITLED-WORK
ABSTRACT
SHAPE
ADJECTIVE
COLOR
DISEASE
TEXT
NARRATIVE*
GENERAL-INFO
DEFINITION
USE
EXPRESSION-ORIGIN
HISTORY
WHY-FAMOUS
BIO
ANTECEDENT
INFLUENCE
CONSEQUENT
CAUSE-EFFECT
METHOD-MEANS
CIRCUMSTANCE-MEANS
REASON
EVALUATION
PRO-CON
CONTRAST
RATING
COUNSEL-ADVICE

Figure 2. Portion of Webclopedia QA Typology.

BEST COPY AVAILABLE

To create the QA Typology, we analyzed 17,384 questions and their answers (downloaded from answers.com); see (Gerber, 2001). The Typology contains 94 nodes, of which 47 are leaf nodes; a section of it appears in Figure 2.

Each Typology node has been annotated with examples and typical patterns of expression of both Question and Answer, as indicated in Figure 3 for Proper-Person.

Question examples	**Question templates**
Who was Johnny Mathis' high school track coach?	who be <entity>'s <role>
Who was Lincoln's Secretary of State?	
Who was President of Turkmenistan in 1994?	who be <role> of <entity>
Who is the composer of Eugene Onegin?	
Who is the CEO of General Electric?	

Actual answers	**Answer templates**
Lou Vasquez, track coach of…and Johnny Mathis	<person>, <role> of <entity>
Signed Saparmurad Turkmenbachy [Niyazov], president of Turkmenistan	<person> <role-title*> of <entity>
...Turkmenistan's President Saparmurad Niyazov…	<entity>'s <role> <person>
...in Tchaikovsky's Eugene Onegin...	<person>'s <entity>
Mr. Jack Welch, GE chairman...	<role-title> <person> ... <entity> <role>
...Chairman John Welch said ...GE's	<subject>|<psv object> of related role-verb

Figure 3. Portion of QA Typology node annotations for Proper-Person.

5. Parsing

Some answers returned by a youthful Webclopedia showed the need to ensure that the answer found is of the right kind semantically:

Q: Where are zebras most likely found? — A: in the dictionary

Q: Where do lobsters like to live? — A: on the table / at the same rate as regular lobsters

and in the right range numerically:

Q: How many people live in Chile? — A: nine

We use CONTEX, a parser that is trained on a corpus to return both syntactic and semantic information, to help.

CONTEX is a deterministic machine-learning based grammar learner/parser that was originally built for MT (Hermjakob, 1997; Hermjakob and Mooney, 1997), where a smaller version of CONTEX (lexically restricted English) reached a labeled precision rate of 89.8% when trained on 256 sentences. Over the past few years it has been extended over the past years to handle deployment on new languages, including Japanese and Korean (Hermjakob, 2000). The Japanese version of the parser, trained on 4096 sentences and tested on lexically unrestricted sentences, achieves 91.4% labeled precision and 91.1% labeled recall for parse trees with a word level granularity, and a bunsetsu level dependency accuracy rate of 84.5%. For English, CONTEX parses of unseen sentences measured 87.6% labeled precision and 88.4% labeled recall, after being trained on 2048 sentences from the Penn Treebank in March 2000. The robustness and the fact that the parser produces a complete parse tree for every test sentence, makes it very useful for Webclopedia.

CONTEX works as follows. As with statistical systems, the grammar learning system also induces its rules from training data; however, it makes better use of linguistic knowledge and other knowledge resources. When presented with a set of parse trees, the learning system automatically derives the sequence of Shift-Reduce parsing operations required to produce each tree. To determine which specific action to take at any point, it considers features of the left and right contexts of the current word. These features include words, parts of speech, lexical and semantic features, etc. In cases of ambiguity, it asks the trainer to identify which feature(s) to pay attention to. Viewing ambiguity as a decision making problem, the system builds a variant of a decision tree to handle the ambiguity in future, using the feature(s) within the context as well as background knowledge in the form of lexicons, ontologies, and any results from topic detection, etc. The appropriate features are indicated manually, by the trainer, if he or she decides they are needed. The decision structure however differs from a traditional grammar in two ways: (1) it is more, in the sense that it does not only provide a space of possible analyses, but in fact selects what it believes is the best analysis, and (2) it has a very operational character in that it directly drives the shift-reduce parser. The grammar as represented by the decision structure therefore has a somewhat different character from the traditional static grammar resource.

Manual guidance allows CONTEX to require far fewer treebanked sentences for training. CONTEX derives much of its strength from the integration of different types of background knowledge, even if those knowledge resources are incomplete. In this way it is a good example of the hybridization of statistical and symbolic techniques. Machine learning algorithms automatically select the most relevant features that best support specific run time parse decisions. This approach employs human and machine each to best advantage: linguists are good at parsing individual sentences, but less good at keeping all the complexity and generalization of a full grammar under control, while machines are excellent at managing and generalizing large sets of individual data points. The result is a rapid traversal of the learning space toward a robust, wide-coverage grammar and parser.

Webclopedia required four extensions to CONTEX.

First, the grammar had to be extended to handle questions. This was achieved by adding approx. 250 manually parsed questions to the Penn Treebank, on which the system's English grammar has been trained. Of these questions, 100 were obtained from NIST's TREC-8 QA corpus and 150 from elsewhere.

Second, the semantic type ontology in CONTEX was extended, both to include QA types and to include many more Objects from our ontology SENSUS. It now contains about 10,000 nodes.

Third, the results of BBN's IndentiFinder locating proper names had to be taken into account.

Fourth, the parse tree output had to be augmented to carry question-related information. The semantic type of the desired answer, as determined by CONTEX, we call the Qtarget. CONTEX returns a ranked list of Qtargets, in order of specificity, drawn from its ontology. For example, the expression

(((c-date) (c-temp-loc-with-year)) ((eq c-temp-loc)))

indicates that the system should try to match a specific date or specific year (both first choice) over a more general temporal expression like "after the war".)

Beside the Qtargets that refer to concepts in CONTEX's concept ontology (see first example in Figure 4), Qtargets can also refer to part of speech labels (see first example), to constituent roles or slots of parse trees (see second and third examples), and to more abstract nodes in the QA

Typology (see later examples). For questions with the Qtargets Q-WHY-FAMOUS, Q-WHY-FAMOUS-PERSON, Q-SYNONYM, and others, the parse tree also provides a slot Qargs that contains additional information helpful for matching (see final examples).

Semantic ontology types (I-EN-CITY) **and part of speech labels** (S-PROPER-NAME):

What is the capital of Uganda?
QTARGET: (((I-EN-CITY S-PROPER-NAME)) ((EQ I-EN-PROPER-PLACE)))

Parse tree roles:

Why can't ostriches fly?	QTARGET: (((ROLE REASON)))
Name a film in which Jude Law acted.	QTARGET: (((SLOT TITLE-P TRUE)))

QA Typology nodes:

What are the Black Hills known for?	Q-WHY-FAMOUS
Who was Whitcomb Judson?	Q-WHY-FAMOUS-PERSON
What is Occam's Razor?	Q-DEFINITION
A corgi is a kind of what?	Q-DEFINITION
What is another name for nearsightedness?	Q-SYNONYM
Aspartame is also called what?	Q-SYNONYM
Should you exercise when you're sick?	Q-YES-NO-QUESTION
True or false: Chaucer was an actual person.	Q-TRUE-FALSE-QUESTION

Qargs for additional information:

Who was Betsy Ross? QTARGET: (((Q-WHY-FAMOUS-PERSON))) QARGS: (("Betsy Ross"))
How is "Pacific Bell" abbreviated? QTARGET: (((Q-ABBREVIATION))) QARGS: (("Pacific Bell"))
What are geckos? QTARGET: (((Q-DEFINITION))) QARGS: (("geckos" "gecko") ("animal"))

Figure 4. QA-related information, returned in the parse tree of the question.

6. Answer Matching

Given the instantiated QA patterns, the Qtargets and Qargs lists, and the potential answer-bearing text segments (also parsed by CONTEX), the Matcher module performs three attempts to pinpoint the answer:

- match QA patterns,
- match Qtargets and Qargs,
- (if all else fails) move a word-level window across the (unparsed) text, scoring each position.

The window scoring function is as follows:

$$\text{Score} = (500 / (500+w))*(1 / r) * \Sigma[(\Sigma I^{1.5*e*u*b*q})^{1.5}]$$

Factors:

- *w*: window width (modulated by gaps of various lengths: "white house" ≠"white car and house")
- *r*: rank of Qtarget in list returned by CONTEX
- *I*: window word information content (inverse log freq)
- *q*: # different question words, and specific rewards (bonus *q=3.0*)
- *e*: penalty for question word expansion using WordNet synsets (*e=0.8*)
- *b*: boosting for main verb match, target words, proper names, etc. (*b=2.0*)
- *u*: (value 0 or 1) indicates whether a word has been "subsumed" by the Qtarget model and should not contribute (again) to the score. For example, "In what year did Columbus discover America?" the subsumed-words are {what, year}.

Unless required, we will not try to develop a more sophisticated scoring function, preferring to focus on the modules that employ information 'deeper' than the word level.

7. Experiments and Results

We entered the TREC-9 short form QA track, and received an overall Mean Reciprocal Rank score of 0.318, which put Webclopedia in essentially tied second place with two others. (The winning system far outperformed all the others.)

A sample analysis of the relative performance of the three modules appears in Table 2. It is clear that the QA patterns made only a small contribution, and that the Qtarget made by far the largest contribution. Interestingly, the word-level window match lay somewhere in between.

Date	IR hits	QA pattern	Qtarget	Window	Total
6/17	78.1	05.5	26.2	10.4	30.3

Table 2. Correct answers attributable to each module.

We are pleased with the performance of Qtargets. They indicate the value of trying to locate the desired semantic type from the meaning of the question. Together with the parse structure, they also help with pinpointing the answer closely: our average answer window length was approx. 25 bytes.

We are not however satisfied with the manually built QA patterns. First, it is too difficult and takes too long to build them by hand (the 500 we have were assembled by simply combining approx. 25 question patterns with 25 answer patterns). Second, the patterns are not robust in the face of small variations of phrasing. We aim instead to build the QA patterns automatically.

8. References

Bikel, D., R. Schwartz, and R. Weischedel. 1999. An Algorithm that Learns What's in a Name. *Machine Learning—Special Issue on NL Learning*, 34, 1–3.

Choi, F.Y.Y. 2000. Advances in independent linear text segmentation. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics* (ANLP-NAACL-00), 26–33.

Fellbaum, Ch. (ed). 1998. *WordNet: An Electronic Lexical Database.* Cambridge: MIT Press.

Gerber, L. 2001. A QA Typology for Webclopedia. In prep.

Hearst, M.A. 1994. Multi-paragraph segmentation of expository text. *Proceedings of the* Annual Meeting of the Association for Computational Linguistics (ACL-94).

Hermjakob, U. 1997. *Learning Parse and Translation Decisions from Examples with Rich Context.* Ph.D. dissertation, University of Texas at Austin. file://ftp.cs.utexas.edu/pub/mooney/papers/hermjakob-dissertation-97.ps.gz.

Hermjakob, U. and R.J.Mooney. 1997. Learning Parse and Translation Decisions from Examples with Rich Context. In *35th Proceedings of the Conference of the Association for Computational Linguistics (ACL)*, 482–489. file://ftp.cs.utexas.edu/pub/mooney/papers/contex-acl-97.ps.gz.

Hermjakob, U. 2000. Rapid Parser Development: A Machine Learning Approach for Korean. In *Proceedings of the North American chapter of the Association for Computational Linguistics* (NAACL-2000). http://www.isi.edu/~ulf/papers/kor_naacl00.ps.gz.

Moldovan, D., S. Harabagiu, M. Pasca, R. Mihalcea,, R. Girju, R. Goodrum, and V. Rus. 2000. The Structure and Performance of an Open-Domain Question Answering System. In *Proceedings of the Conference of the Association for Computational Linguistics* (ACL-2000), 563–570.

Srihari, R. and W. Li. 2000. A Question Answering System Supported by Information Extraction. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics* (ANLP-NAACL-00), 166–172.

Witten, I.H., A. Moffat, and T.C. Bell. 1994. Managing Gigabytes: Compressing and indexing documents and images. New York: Van Nostrand Reinhold.

Question Answering by Passage Selection
(MultiText Experiments for TREC-9)

C. L. A. Clarke G. V. Cormack D. I. E. Kisman T. R. Lynam

Computer Science, University of Waterloo, Waterloo, Ontario, Canada
mt@plg.uwaterloo.ca

1 Introduction

MultiText has participated in TREC each year since TREC-4 [3, 2, 7, 8, 6]. For TREC-9 we concentrated our efforts on the question answering (QA) track and also submitted runs for the Web track.

The MultiText system incorporates a unique technique for arbitrary passage retrieval, which was used in all of our TREC-9 experiments. The technique efficiently locates high-scoring passages, where the score of a passage is based on its length and the weights of the terms occurring within it. Passage boundaries are determined by the query, and can start and end at any term position. If a document ranking is required, the score of a document is computed by combining the scores of the passages it contains. Versions of the technique have been described in our previous TREC papers and elsewhere [4]. Our TREC-8 paper [6] provides a concise overview of the current version.

Our TREC-9 QA experiments extended our TREC-8 work. For TREC-8 we simply stripped stopwords from the question, applied the passage retrieval technique, and submitted 250 or 50 bytes centered at each of the top five passages. Considering that no use was made of the structure of the question or the meaning of words within it, relatively good results were achieved. For TREC-9 we applied both question pre-processing, to a generate a query that is more likely to retrieve passages that contain the answer, and passage post-processing, to select the best 250 or 50 byte answer from within a passage. Both the pre-processor and post-processor make use of a question parser, which generates the query to be executed against the target collection and a set of selection rules that are used to drive a set of pattern matching routines in the post-processor.

In addition, we participated in all aspects of the Web Track, submitting runs over both the 10GB and 100GB collections, 10GB runs incorporating link information, and 10GB runs using both title-only and title-description queries.

2 Question Answering Track

Our question answering system consists of three main components (Figure 1). The parsing component performs the pre-processing of questions, feeding the resulting queries and selection rules to the passage retrieval component and the passage selection component. The passage retrieval component executes the queries over the target corpus, retrieving the ten best passages. The passage selection component applies the selection rules to the passages to produce a ranked set of five 50- or 250-byte answers.

BEST COPY AVAILABLE

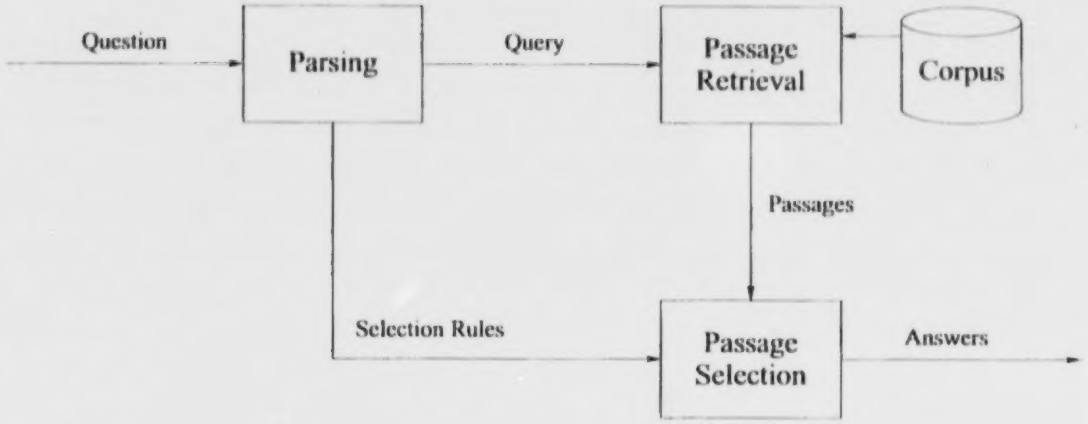


Figure 1: Overview of QA processing

2.1 Passage Retrieval for Question Answering

The MultiText passage retrieval algorithm is used to identify locations in documents where answers are likely to occur. For each question, ten passages are retrieved and passed to the passage selection component for analysis. These ten passages consist of the highest-scoring passages where no two passages are taken from the same document.

For passage retrieval purposes, each document D in the corpus is treated as an ordered sequence of words:

$$D = d_1\ d_2\ d_3 ... d_m.$$

At each word position $(1...m)$ various terms are indexed indicating information about the word. These indexed terms include the word itself, the stemmed word and in some cases a token indicating that the word's position corresponds to the start of a name, a number, a monetary value, or other potential answer value.

A query is treated as a set of terms:

$$Q = \{q_1,\ q_2,\ q_3, ...\},$$

where each term is a word, a phrase, a truncated word, a word with stemming applied or a disjunction of terms. Document and query terms are matched using the expected semantics. For example, the query used for question 425 ("How many months does a normal human pregnancy last?") is

```
months normal human pregnancy "$last" <duration>
```

The term `"$last"` indicates that "last" should be matched under stemming (matching "lasts", "lasting", etc.). The term `<duration>` matches positions in the document where possible time values have been identified during indexing of the corpus.

An *extent* (u, v), with $1 \leq u \leq v \leq m$ is used to represent a subsequence of D beginning at position u and ending at position v:

$$d_u\ d_{u+1}\ d_{u+2} ... d_v.$$

An extent (u, v) *satisfies* a term set $T \subseteq Q$ if the subsequence of D defined by the extent matches all the terms from T. An extent (u, v) is a *cover* for T if (u, v) satisfies T and the subsequence corresponding to (u, v) contains no subsequence that also satisfies T. That is, there does not exist an extent (u', v') with either $u < u' \leq v' \leq v$ or $u \leq u' \leq v' < v$ that satisfies T. The MultiText System uses a fast algorithm to compute covers for all subsets of Q over all documents in a collection [4].

Passages are assigned scores based on their lengths and on the weights assigned to the query terms they match. A term t is assigned an IDF-like weight:

$$w_t = \log(N/f_t),$$

where f_t is the number of times that t is matched in the corpus and N is the sum of the lengths of all the documents in the corpus. A standard IDF weight is not used since in-document frequencies are not stored in the MultiText index.

The weight assigned to a set of terms $T \subseteq Q$ is the sum of the weights assigned to each term in T:

$$W(T) = \sum_{t \in T} w_t.$$

If an extent (u, v) is a cover for the term set T then it can be assigned a score combining the length of the extent and the weight of its matching terms:

$$C(T, u, v) = W(T) - |T| \log(v - u + 1). \tag{1}$$

Once the ten highest-scoring extents from distinct documents were determined, the centerpoint of each extent was computed $(u + v)/2$ and a 200-word passage centered at this point was retrieved from the corpus. These 200-word passages were then used for final passage selection.

The ability of the algorithm to locate potential answers is illustrated by Figure 2. The figure is based on the top passage retrieved for each of the 200 TREC-8 questions. The queries used to retrieve these passages were generated using our TREC-9 parser, described in Section 2.2. A 50-byte window was slid across each passage a byte at a time, and for each location of the window a regular expression was used to check for an answer to the corresponding question. These regular expressions were taken from the QA evaluation script distributed to track participants by Ellen Voorhees. For each possible location of the window's center relative to the center of the retrieved passage, the figure plots the number of questions for which an answer was found in the window. The graph has an obvious spike within ±100 bytes of the centerpoint.

2.2 Parsing Questions

The parser has two functions: 1) to generate better queries so that the passage retrieval engine can generate the best candidate passages, and 2) to generate selection rules so that the post-processor can select the best 50-byte or 250-byte answers from the passages.

As a baseline for comparison, we use our simple but effective TREC-8 technique. For this baseline, query terms are exactly those in the question, except for stop words, with no stemming or expansion of any sort. The post-processor merely truncates or expands the top five retrieved passages to form the run. After TREC-8 we observed a number of shortcomings in this baseline technique, which we aim to address with the parser.

In the baseline, stopwords like "how", "when", "first" are eliminated, sometimes eliminating the crux of the question at the same time. In the majority of cases the eliminated words are useless

BEST COPY AVAILABLE

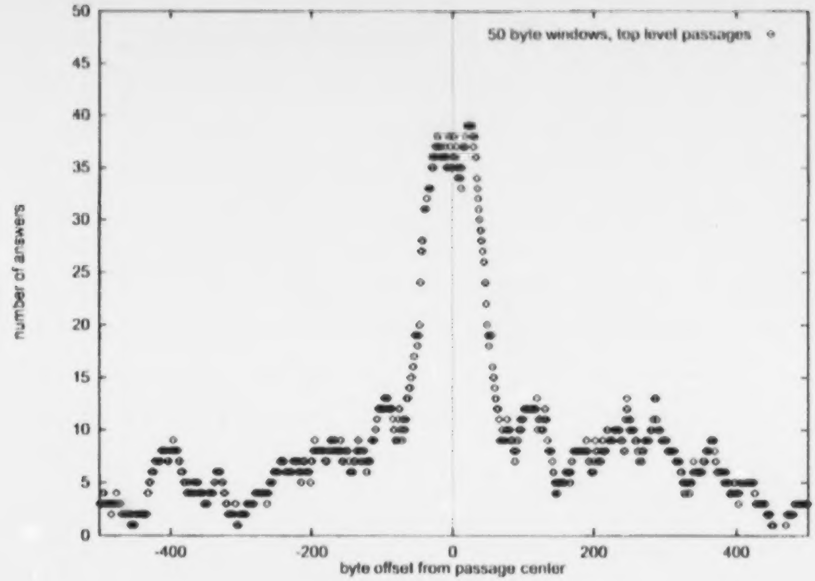


Figure 2: Answer location relative to passage center (TREC-8 data)

as search terms, but in a significant minority of cases the eliminated words would have been useful, and in many other cases the eliminated words contain potentially valuable information about the nature of the question.

In the baseline, words are not stemmed at all. Our experience with the Adhoc, Web, and VLC tasks in previous TREC experiments convinced us that stemming compromises early precision, which is all-important for question answering. Nonetheless, a variant of an exact question word would have often been a better search term. For example, given the question "When did John Kennedy die?", the variant "died" is more likely to appear in the answer than "die". Synonyms like "killed", "shot", or "assassinated" may also glean correct answers. However, in preliminary experiments we found that naive thesaurus expansion, like stemming, compromised early precision, and was overall a detriment to question answering.

We and others [9] have observed that many of the questions can be categorized (Who, What, Where, How far, How many, etc) and that answers to questions of a particular category often contained particular terms. For example, answers to "How far" questions almost certainly contain a number and a unit of distance. Answers to "How many" questions also contain a number and a unit, but the unit term depends on the question rather than the question category alone. For example, the answer to "How many angels can dance on the head of a pin?" almost certainly contains a number followed by "angel" or "angels".

Some categories contain more subtle information about the form of the answer. The answer to "What is the capital of Uraguay?" almost certainly contains the name of a city, or, to be more specific, a capital city. This particular question is common enough in the TREC-8 questions that one might consider handling "capital" as a special case, but we wished to find a more general solution. "What river flows through Kansas City?" is more subtle still. With appropriate analysis one could deduce that the answer is a river, and enumerate the possible answers, or one could make

BEST COPY AVAILABLE

use of the verb "flows" to deduce that a likely form of the answer is the name of the river followed by some conjugation of "flow".

The parser attempts to infer some of this information from the question and use it to generate the queries and selection rules. The queries and the selection rules capture the results of the parsing in different forms. The queries consist of terms that are likely to appear close to answer. The selection rules explicitly contain category and part-of-speech information, as well as lexical patterns that cannot be matched by the passage retrieval system.

2.2.1 Implementation of the Parser

After considering a number of approaches to question analysis, we developed our own context-free grammar and parser. We considered using shallow (finite-state) patterns but rejected them because: 1) it appeared that the patterns would be as complex as a context-free grammar; 2) we were concerned about over-fitting our analysis to the training data; and 3) we wanted the structure of a parse tree as a framework for question analysis. We also investigated existing natural language parsers, in particular Cooper's CPSG [5] and the Link Grammar Parser [10]. Neither appeared to parse the training questions well, and in both cases we estimated that building an interface for the parser and its output was as complex as parsing the question directly.

Part-of-speech analysis is a necessary complement to context-free parsing, as it is not possible to write a grammar whose vocabulary incorporates all possible words in the question. Instead, generic words like `<verb>`, `<noun>`, etc. are recognized by a separate component. We did not find a publicly available part-of-speech tagger suitable to our needs. Instead we used WordNet to determine whether each word of the question was likely a noun, verb, adjective, or adverb. Parts of speech not recognized by WordNet (articles, pronouns, prepositions) were explicitly enumerated in the grammar. We used a simple strategy suggested in the WordNet documentation [1]: We assumed that the probability of a word being a particular part of speech was proportional to the number of senses listed for that part-of-speech. For example, the word "head" has 30 noun senses, 6 verb senses, 2 adjective senses, and no adverb senses, so we assign probabilities 0.79, .16, .05, and ϵ to these senses. In addition, explicit words in the grammar (like articles, pronouns, and important other words) are given their own categories with high probabilities.

The parser uses Earley's algorithm to determine all possible parses and selects the one that is most probable; the probability of a rule is the product of the probabilities of the terms on its right side, multiplied by an optional weight factor associated with the rule. For example, the probability of matching the rule

```
    NOUNPHRASE -> <adjective> NOUN 0.9
```

is the probability of matching `<adjective>` times the probability of matching `NOUN` times 0.9. The grammar contains only 80 production rules (see appendix).

2.2.2 Analysis of the Parse Tree

The most probable derivation tree is walked by an attribute evaluator. The evaluator accumulates each of the important words and its part of speech (noun, verb, adjective, or adverb) and attempts to identify the subject and predicate of the question. Quoted phrases and capitalized sequences of words are identified. The category of the question is identified for various phrasings. For example "Name the capital of Uruguay", "What is the capital of Uruguay?", and "What is Uruguay's capital?" all yield the same result. Special forms involving "How" are recognized as specific categories. Finally, an attribute "instanceof" is computed which identifies a hypernym of the answer. For the

BEST COPY AVAILABLE

examples above, this attribute is “capital”. The parse tree and the computed attributes are written to an intermediate file which is processed by the query generator.

The query generator uses the attributes to formulate queries and selection rules. It deduces more specific answer categories using the “instanceof” and other attributes as well as its knowledge of a few specific words. For example, if the “instanceof” attribute is one of “value”, “debt”, “earnings”, etc. the answer category is assumed to be “money”. If the question category is “How long” the answer category may be “distance” or “time interval”. For each possible answer category, the generator may add compound terms that retrieve likely answers. For example, the answer category “time interval” causes a term like `("hours"+"minutes"+"weeks"+ ... +"aeons")` to be added to the query. The generator further generates a selection rule that would identify a number followed by one of these terms.

Quoted or capitalized phrases were added as query terms in addition to each of their constituent words. Possessive adjectives were expanded. For example, queries for the capital of Uruguay would all include the term `("Uruguay's"+"of Uruguay"+"of the Uruguay")`. Verbs were stemmed and irregular verbs were expanded to include all conjugations. Nouns, adjectives, and adverbs were not stemmed. Pronouns, articles and prepositions were omitted from the query. We had intended to use WordNet and a number of special-purpose dictionaries to expand a large number of “instanceof” attributes. Unfortunately, time did not permit.

All of the attribute and category information was written to yet another intermediate file, which was read by the post-processor. A separate pattern file translated some answer categories and “instanceof” attributes into lexical patterns.

2.3 Passage Selection

For each question, the passage selection post-processor receives a list of ten ranked passages from the retrieval engine and category information from the parser. In addition, the post-processor consults external databases containing lists of countries, states, cities, proper names, etc. Post-processing then proceeds with the following steps:

1) Determine the answer category from the parser output.
2) Scan the passages for patterns matching the answer category.
3) Assign each possible answer term an initial score based on its rarity.
4) Decrease or increase the the term scores depending on various quality heuristics.
5) Select from the passages the (50-byte or 250-byte) answer that maximizes the sum of the term scores it contains.
6) Set the scores of all terms appearing in the selected answer to zero.
7) Repeat steps 5 and 6 until five answers are selected.

In the first step, the answer categories yielded by the parser are reduced by the pre-processor to one of “Proper” (person, name, company, etc.), “Place” (city, country, date, etc.), “Time” (date, duration, weekday, etc.), “How” (much, many, far, long, etc.), or “Other”. The “How” category includes special subcategories for monetary values, numbers, distances and other measurements.

Next the passages are scanned using the patterns for the given category (step 2). These pattern generally consist of regular expressions with simple hand-coded extensions. For example, the pattern for “Proper” is simply `[^A-Za-z0-9][A-Z][A-Za-z]+[^A-Za-z0-9]`, which matches a capital letter followed by one or more letters surrounded by white space or punctuation. Repeated strings are considered more likely to be the answer, as are strings found in highly ranked passages and strings found near the center of passages.

BEST COPY AVAILABLE

The third step is to assign a score to the term, using the formula

$$c_t \log(N/f_t),$$

where f_t is the number of times the term appears in the corpus, N is the sum of the lengths of all the documents in the corpus, and c_t is the number of retrieved passages in which the term appears.

The fourth step modifies the scores using a number of heuristics. First, the score is modified according to its distance from the center of the the retrieved passage. The score is multiplied by $1 - \mathit{distance}/1000$, lowering the scores for terms that are farther from the center. Then the score is modified depending on the rank of the passage in which it was found. It is multiplied by $1 - \mathit{rank}/1000$, giving a lower score to terms from passages ranked lower by the retrieval engine. The term score is also modified using additional patterns specific to the answer category. Some patterns are "boosters" which increase the score, while some are "reducers" which decrease the score. For example for a "Proper" answer category, the occurrence of "Mr." before the term is a booster, whereas being part of a "Date" pattern is a reducer. The exact values of the boosters and reducers were determined and adjusted manually using the training data.

For each 50- or 250-byte substring of the passages, its score is simply the sum of the scores of the terms within it. The best answer is the substring of the required length with the highest score. This answer is selected for inclusion in the run, and the scores of all terms appearing in it are reduced to zero (step 6). The next best answer is then selected, and this process repeats until five answers are generated (step 7). The purpose of reducing the term scores to zero after selection is to eliminate duplication. However, there is a still a risk that part of the answer may appear in an incorrect passage and the reduction in weight might cause the correct answer to be missed.

Consider the question "Who is the leader of India?" (question 215). The highest-scoring terms and their scores (scaled by 1/1000) are "Sikhs"(27), "Vishwanath"(22), "Pratap"(20), "Tamil"(16), "Farooqui"(8), "Wire"(7), "Nadu"(5), "Madras"(3), "Punjab"(3),"Indian"(2), "Velupillai"(1.6), "urges"(1.5), "Hindu"(1.1), "Prabhakaran"(9.8), "Gandhi"(0.7), "Dixit"(0.7), "Participation"(0.7), and "Singh"(0.6). The top five 50-byte passages returned by the postprocessor are:

1) `. Indian Prime Minister Vishwanath Pratap Singh f`
2) `. Front. INDIA LEADER URGES SIKHS' PARTICIPATION.`
3) `PUNJAB PEACE. From Times Staff and Wire Reports.`
4) `unist Party of India) leader, Mr M. Farooqui. "Bu`
5) `d Monday. J.N. Dixit said Velupillai Prabhakaran,`

2.4 Results

The officials results from our QA runs are summarized in Figure 3. Two of the runs (`uwmt9qas0` and `uwmt9qal0`) were generated by the method described in Section 2.3. The other two (`uwmt9qas1` and `uwmt9qal1`) were generated by a similar technique that used a different (and less successful) approach to pattern matching.

Along with the official strict and lenient mean reciprocal ranks, in its last column the figure lists the results of our own internal judging of the 250-byte runs, which was undertaken immediately after the QA runs were submitted. Since we could only submit two official runs, we performed our own judging to examine the individual effects of the various QA processing components. The results of are presented in Figure 4.

The values in Figure 4 are for questions 201-700 only, since the remaining questions are rephrasing of previous questions. The first row of the figure gives the mean reciprocal rank for a baseline

BEST COPY AVAILABLE

run id	length (bytes)	strict official judgements	lenient official judgements	MultiText judgements
`uwmt9qas0`	50	0.321	0.339	
`uwmt9qas1`	50	0.257	0.264	
`uwmt9qal0`	250	0.456	0.475	0.486
`uwmt9qal1`	250	0.460	0.465	0.456

Figure 3: Mean reciprocal ranks over all QA questions

	strict official judgements	lenient official judgements	MultiText judgements
baseline			0.414
parser-generated queries			0.464 (+12%)
`uwmt9qal0`	0.464	0.471	0.502 (+21%)
`uwmt9qal1`	0.457	0.460	0.472 (+14%)

Figure 4: Mean reciprocal ranks for 250 byte runs over QA questions 201-700 only

run that duplicates the method used for our TREC-8 QA submissions. The queries used for this run were obtained by stripping the stopwords from the questions, and answer selection consisted of truncating or extending the top five retrieved passage to the appropriate length. The next row shows the effect of the parser. For this run, the queries were generated by the parser, but passage selection post-processing was not preformed. As with the baseline run, the answer was produced by truncation or extension of the retrieved passages. The last two rows give the mean reciprocal rank over questions 201-700 using both the official judgements and our judgements. For our best 250-byte run, the use of the question parser gave a 12% improvement over the baseline technique, and passage selection gave a further 8% improvement, for an overall improvement of 21%.

3 Web Track

We submitted six runs for the small (10GB) Web track and three runs for the large (100GB) Web track. Document ranking for all of the runs was based on the same version of our *cover density ranking* algorithm that we used for our TREC-8 experiments [6]. New for TREC-9 was the use of a 4-gram index for the small Web track and some limited use of document links. Cover density ranking is an extension of the passage retrieval technique described in section 2.1, in which the score of a document is computed from the scores of the passages it contains.

Figure 5 summaries our submissions for the small Web track. Three runs use only page content for ranking. The first (`uwmt9w10g0`) uses words from the title only; the second (`uwmt9w10g1`) uses words from both the title and description. For both runs, query terms were generated by normalizing the words to lower case and eliminating stopwords. The terms were not stemmed since it was our experience in earlier TREC Web experiments that stemming adversely effects the precision in the top 20 or so documents. The third content-only run (`uwmt9w10g4`) is based on the title only, but in contrast to the other runs, this run uses 4-gram indexing for retrieval.

Web queries and documents often contain significant spelling errors, and our use of 4-grams was planned to address this problem. For indexing, each word in each document was split into

BEST COPY AVAILABLE

run id	run description	avg. prec.	prec. @ 10
`uwmt9w10g0`	title-only, content-only	0.165	0.238
`uwmt9w10g1`	title-description, content-only	0.133	0.260
`uwmt9w10g2`	title-only, content-link (from `uwmt9w10g0`)	0.163	0.236
`uwmt9w10g3`	title-description, content-link (from `uwmt9w10g1`)	0.134	0.262
`uwmt9w10g4`	4-gram, title-only, content-only	0.181	0.240
`uwmt9w10g5`	4-gram, title-only, content-link (from `uwmt9w10g4`)	0.179	0.240

Figure 5: Small Web track results

overlapping 4-grams. For example, the word "tomato" would be split into the terms "toma", "omat" and "mato". Each word was split individually; 4-grams did not span multiple words or contain punctuation or white space. Cover density ranking requires that the word position of each term be available in the index, and in this case all 4-grams generated by a word were treated as occurring at the word's position. Words consisting of less than 4 characters were indexed directly.

We were surprised at the relative difference between the title-only run using words (`uwmt9w10g0`) and the title-only run using 4-grams (`uwmt9w10g4`). The 4-gram run achieved a 9% better average precision but similar precision at 5-20 documents. We had expected the 4-gram run to exhibit better performance on those queries that contained significant spelling errors, and this hypothesis was confirmed in a limited way. However, only a few of the titles actually contained spelling errors, and most of the performance difference appears to be due the matching of morphological variants by the 4-gram queries. This result is in contrast to our earlier experience with stemming.

We were also surprised at the relative difference between the title-description run (`uwmt9w10g1`) and the corresponding title-only run (`uwmt9w10g0`), with the title-only run achieving a 24% better average precision.

The remaining three small Web track runs (`uwmt9w10g2`, `uwmt9w10g3`, and `uwmt9w10g5`) represent our first attempt to use link information for TREC experiments. Each run was generated from one of the content-only runs by using link information in an additional re-ranking step. The re-ranking had little impact on retrieval effectiveness.

The table of Figure 6 summaries our submissions for the large Web track. Our three TREC-9 submissions repeat our submissions for the TREC-8 large Web track and the figure directly compares the two years. The runs may also be compared with our small Web word-based, title-only run (`uwmt9w10g0`) which uses similar queries and essentially the same ranking method. All three large Web runs use the topic words mapped to lower case with stopwords eliminated. The first run (`uwmt9w100g0`) uses these terms without change. The second run (`uwmt9w100g1`) extends these terms with a simple stemmer for plurals. The third run (`uwmt9w100g2`) uses only the three terms having the greatest term weights (w_t). The difference between our TREC-8 and TREC-9 results is quite dramatic considering that the only difference is in the selection of the query subset for judging and the actual judging itself.

	TREC-9		TREC-8	
run id	avg. prec.	prec. @ 10	avg. prec.	prec. @ 10
uwmt9w100g0	0.351	0.454	0.478	0.586
uwmt9w100g1	0.328	0.427	0.487	0.580
uwmt9w100g2	0.336	0.427	0.470	0.590

Figure 6: Large Web track results

References

[1] Richard Beckwith, George Miller, and Randee Tengi. Design and implementation of the WordNet lexical database and searching software. Included in wordNet software distribution. See `http://www.cogsci.princeton.edu/~wn/`.

[2] Charles L. A. Clarke and Gordon V. Cormack. Interactive substring retriveal. In *Fifth Text REtrieval Conference (TREC-5)*, pages 295–304, Gaithersburg, Maryland, November 1996.

[3] Charles L. A. Clarke, Gordon V. Cormack, and Forbes J. Burkowski. Shortest substring ranking. In *Fourth Text REtrieval Conference (TREC-4)*, pages 295–304, Gaithersburg, Maryland, November 1995.

[4] Charles L. A. Clarke, Gordon V. Cormack, and Elizabeth A. Tudhope. Relevance ranking for one to three term queries. In *Fifth RIAO Conference*, pages 388–400, Montreal, June 1997. A version of this paper will appear in *Information Processing and Management*, 2000.

[5] R. Cooper. *Classification-based Phrase Structure Grammar — An Extended Revised Version of HPSG*. PhD thesis, University of Edinburgh, 1990.

[6] G. V. Cormack, C. L. A. Clarke, C. R. Palmer, and D. I. E. Kisman. Fast automatic passage ranking. In *Eighth Text REtrieval Conference (TREC-8)*, Gaithersburg, Maryland, November 1999.

[7] Gordon V. Cormack, Charles L. A. Clarke, Christopher R. Palmer, and Samual S. L. To. Passage-based refinement. In *Sixth Text REtrieval Conference (TREC-6)*, pages 303–319, Gaithersburg, Maryland, November 1997. A version of this paper will appear in *Information Processing and Management*, 2000.

[8] Gordon V. Cormack, Christopher R. Palmer, Michael Van Biesbrouck, and Charles L. A. Clarke. Deriving very short queries for high precision and recall. In *Seventh Text REtrieval Conference (TREC-7)*, pages 121–132, Gaithersburg, Maryland, November 1998.

[9] John Prager, Eric Brown, Anni Coden, and Dragomir Radev. Question-answering by predictive annotation. In *23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 184–191, Athens, July 2000.

[10] Davy Temperley, Daniel Sleator, and John Lafferty. The link grammar parser. See `http://www.link.cs.cmu.edu/link`.

Appendix - The Grammar

```
START -> NUM SENTENCE

SENTENCE -> Capital S .9 | S

S -> VP
   | NP .8
   | name NP
   | what TOBE the NAME AJPREP NP 1.2
   | NP VP
   | AV DID NP VP
   | NP DID NP VP
   | NP TOBE VV
   | NP TOBE NP
   | AJ TOBE NP
   | AJP TOBE NP

NAME -> name | names | term | terms | abbreviation | acronym

DID -> did | does | will | were | was | do | has | had | have

TOBE -> is | was | will be | were | are | have | has
      | contains | Punc-' s

VP -> VV
    | VV NP
    | VV NP AVP

VV -> V
    | VV AVP
    | VV CONJ V

NP -> NN
    | NP COMMA AJP COMMA
    | NP CONJ NN

CONJ -> and | or | Punc-,

NN -> N
    | ART N

N -> Noun | PRON | AJ N | NUM | STR | Capital Noun | Capital PRON
   | Gerund Noun | name | Initial Noun | Acronym Noun | Capital a | US

US ->  Acronym us 1.1 |
    | Initial u Initial s 1.1
    | Capital united Capital states 1.1 | Acronym usa 1.1
    | Initial u Initial s Initial a 1.2

AJ -> Adjective
    | Noun | Initial Noun | Capital Noun | Acronym Noun
    | PRON | Initial PRON | Capital PRON | Acronym PRON
    | Noun Punc-' | Initial Noun Punc-' | Capital Noun Punc-'
    | Noun Punc-' s | Initial Noun Punc-' s | Capital Noun Punc-' s
    | Acronym Noun Punc-' | Acronym Noun Punc-' s
    | AV Adjective
    | Capital Adjective
    | Initial Adjective
    | Acronym Adjective
    | US
    | NUM

AJP ->  AJPREP NP
     | AJINTRO VP
     | AJINTRO NN VP
     | AJPREP Gerund VP
     | Gerund VP

AVP -> COMMA AV COMMA
     | AVPREP NP
     | AVINTRO NP VP
     | AVINTRO VP

V -> Verb
   | Verb AV  1.1
   | AV V
   | Verb V
   | to V
   | Gerund Verb
   | Capital Verb 0.2
   | Initial Verb 0.2
   | Acronym Verb 0.2

AV -> Adverb | AV Adverb | Capital Adverb 0.2
    | Initial Adverb 0.2 | Acronym Adverb 0.2

AVINTRO -> while | during | after | before | when | like | as | upon

AJINTRO -> who | whom | which | that | whose | Punc-, | to | at | off
```

```
AJPREP -> of | over | by | at | in | between | under
        | from | for | upon | on | with | about
| around | Punc-, | off | up | down | into

AVPREP -> over | by | at | in | to | into | under
        | from | for | via | on | onto | with | about
| around | Punc-, AVPREP | since | during | upon

PRON -> his | her | it | its | their | this
      | that | what | our | us | them | he
      | she | who | whom | hers | which

ART -> the | a | an

NUM -> DIG

STR -> COMMA Punc-" STUFF Punc-" COMMA

DIG -> Number

STUFF -> W | STUFF W

W -> Punc-, | Punc-. | Punc-?
     | ART | AVPREP | AJPREP | AVINTRO | AJINTRO | Punc-, | Capital
     | Acronym | Initial | Noun | Verb | Adverb | Adjective | Number

COMMA ->
      | Punc-,
```

Kasetsart University TREC-9 Experiments

P. Narasetsathaporn, A. Rungsawang
{g4365020,fenganr}@ku.ac.th
Massive Information & Knowledge Engineering
Department of Computer Engineering
Faculty of Engineering
Kasetsart University, Bangkok, Thailand.

Abstract

We add pivoted unique normalization weighting scheme to SMART and use it to run the final experiments. Since SMART produces an inverted file which is larger than 2G limitation on our x86 based Linux machine, we have then to divide small web track test set into several subcollections, index and retrieve, and merge all scores to obtain the final result.

1 Introduction

In our TREC-8 experiments last year, we proposed to mine good candidate terms from the document collection, using "Apriori algorithm" [1], and then use that terms to enhance the original query [3]. We then used the new expanded query to retrieve relevant documents in the collection. From those experiments, we had found that the proposed technique worked well with the FT (Financial Time) collection using some weighting pairs, such as lnc.ntc or lnc.atc, and we got till 19% improvement.

In the TREC-9 experiments this year, we try to use the same technique with the whole small web track documents, but confront with many technical obstacles. Firstly, we have spent very much time to write a robust parser to parse messy data in small web track documents. Secondly, the number of different terms, after removing stopwords and stemming, is so numurous that the Apriori algorithm we use to mine good candidate terms, as well as our own DSIR text retrieval algorithm, have hardly come through in time we have left, though using the most powerful x86 PC based machine (equipped with 512M of RAM) we have in our department.

We then decide to modify the Cornell's SMART version 11.0 so that it can run smoothly on our Linux machine, and add the notable pivoted unique normalization weighting scheme [4] to it. However, another intrinsic operating system problem arises. We cannot index the whole small web track documents in one-shot since SMART will generate an inverted-file image that is larger than the 2G limitation of the x86 based Linux machine. Therefore, we have to split the small web track collection into

BEST COPY AVAILABLE

several sub-collections, index and test those sub-collections with SMART, merge all subsequent results to get the final top-1000 scores.

The rest of our report gives more detail about what we do during TREC-9 period. Sections 2 describes new weighting scheme that we add to the original SMART version 11.0. Sections 3 gives more detail about the merging algorithm we use to combine the whole final scores from several runs. Sections 4 provides the final results we obtain, and section 5 concludes this report.

2 Adding New Weight to SMART

Since we found that retrieval results recieved from the classical weighting schemes provided by the original SMART distribution are not as good as we expect, and unfortunately there is no patch for the new well-known weighting schemes, we then decide to mess up some more codes into the original SMART. We have followed the pivot document length normalization weighting scheme introduced by Singhal et al. [4]. This normalization scheme is based upon both normalizing the t_f (term-frequency) factor by the average t_f in the document vector, and the overall vector length by a pivot and a slope factor dependent on the number of unique terms in that document. Based on the underlying t_f factor (which we call the L factor in the SMART tripple weighting notation), and the pivoted unique normalization (which we call the u normalization), we obtain the final weighting scheme, called Lnu weighting in SMART, in the form of:

$$\frac{\frac{1+\log(t_f)}{1+\log(\text{average } t_f)}}{(1.0-\text{slope})*\text{pivot}+\text{slope}*\#\text{of unique terms}} \qquad (1)$$

which the #of unique terms is the amount of term of which t_f is equal to 1, and pivot is the average number of unique terms.

To obtain this weight, we modify the SMART version 11.0 in src/libconvert by adding the function tfwt_triple (L) in weights_tf.c and normwt_unique (u) in weights_norm.c, while during experiments the pivot and slope are read from the spec file. We also add the L and u to tell SMART about this new weight in src/libproc/proc_convert.c

After adding the new weight to SMART, we verify it by running some retrieval experiments. We choose the FR and FBIS, and the topics 401-450 as our test sets. We obtain the results as illustrated in the Table 1 as follows. Results from this test make us quite certain that we do add the correct weight to the old orginal SMART.

Collection	# of relevance	11pt avg precision		Relevant retrieved	
		lnc.ltc	Lnu.ltu	lnc.ltc	Lnu.ltu
FBIS	1667	0.2001	0.2732	1144	1106
FR	206	0.1814	0.2713	170	170

Table 1: Testing results of our modified SMART.

3 Merging Results from Sub-collections

Since the total number of documents in small web track collection, 1,692,096 documents, is quite large for our x86 based Linux machine, the inverted file index produced by SMART becomes messy when it breaks the 2G barrier. We then have to divide the whole collection into several ones, index and retrieve the top-ranked documents, and merge the whole together to get the final ranking scores. There exist several merging algorithms mentioned in the literatures [2, 5]. We investigate and implement four of them, i.e. interleaved merge, raw score merge, normalized score merge, and weighted score merge, and test them using the FR collection. Results from this test shows that the weighted score appraoch gives the best merging results. The weight w, below, is the one we use to combine the whole final ranking scores from several small web track sub-collections in our TREC-9 experiments.

$$w = 1 + |C| * \frac{s - \bar{s}}{\bar{s}} \tag{2}$$

where $|C|$ is the number of sub-collections, s is the collection's score, and $\bar{s}$ is the mean of the collection scores. With this approach, each document is ranked based upon the product of its score and the weight w for its collection [2].

4 Experiments and Results

We first parse all html tags, images, all messy data, and the others, out of the small web track collection. We use every words found in the small web track topics as queries. From several experiments we have performed till the deadline of this final report, we obtain the best final scores when the original small web track collection has been divided into 7 sub-collections, in a round-robin fashion, as concluded in Table 2 as follows.

Sub-collection	Directory	Doc-number
1	WTX001-WTX015	1-251745
2	WTX016-WTX030	251746-485635
3	WTX031-WTX045	485636-730835
4	WTX046-WTX060	730836-976633
5	WTX061-WTX075	976634-1233895
6	WTX076-WTX090	1233896-1474263
7	WTX091-WTX104	1474264-1692096

Table 2: Small web track sub-collections.

We index all the sub-collections seperately, using our own modified SMART running on a x86 based Linux machine, and try with several weighting schemes. We found that the weight Lnu.ltu combination gives the best scores, when pivot and slope parameters in Equation 1 have been set as shown in the Table 3 as follows.

Sub-collection	Pivot	Slope
1	84.53	0.15
2	90.86	0.1
3	83.44	0.05
4	81.05	0.05
5	80.54	0.1
6	82.98	0.25
7	84.15	0.2

Table 3: Pivot and slope parameters.

Using weighted score merging approach, the final result we have is concluded in the Table 4 below. Note that we also give the result obtained from lnc.ltc weight for reference.

Weight	Avg.Precision	Rel.Retrieved	P.@5doc	P.@100doc
lnc.ltc	0.0525	803	0.1080	0.0508
Lnu.ltu	0.1943	1122	0.3360	0.1058

Table 4: KU TREC-9 final small web track results.

5 Conclusion

In our TREC-9 experiment this year, we have not provided any valuable finding to the web track community, but just participate in spirit. Till the last day of this final report deadline, we do hardly come through to get some results. We have spent a lot of time to code a robust parser to extract free text from small web track documents. We lost much time to alter the Apriori algorithm to run with big small web track data, but it has never been converted to give any results. We also face with the intrisic Linux operating system problem when the file size is larger than 2G barrier on the x86 machine.

Since we do not succeed to let the Apriori algorithm convert on the whole small web track data, our claim last year about the query enhancement technique [3] is still not verified. We turn to use our own modified version of SMART to run the experiments. We add the new weight, i.e. pivoted unique normalization, in SMART. Since the problem of 2G file size limitation in our Linux based machine still exists, we then divide the small web track test set into several sub-collections, index and retrieve relevant documents seperately, and use weighted score merging technique to combine the final top-1000 scores.

Ackknowledgement

We would like to thank our MIKE staffs, especailly Hong, Wit, Tew, for their programming support and working spirit. We also thank to Ink who worked so hard with Apriori algorithm last year.

References

[1] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In *Proceedings of the* 20^{th} *VLDB Conference*, Staiago, Chile, 1994.

[2] J.P. Callan, Z. Lu, and W.B. Croft. Searching Distributed Collections with Inference Networks. In *Proceedings of the* 18^{th} *ACM-SIGIR Conference*. ACM Press, 1995.

[3] A. Rungsawang, A. Tangpong, P. Laohawee, and T. Khampachua. Novel Query Expansion Technique using Apriori Algorithm. In E. Voorhees and D. Harman, editors, *Proceedings of the* 8^{th} *Text REtrieval Conference*. NIST Special Publication 500-246, 1999.

[4] A. Singhal, C. Buckley, and M. Mitra. Pivoted Document Length Normalization. In *Proceedings of the* 19^{th} *ACM-SIGIR Conference*. ACM Press, 1996.

[5] E. Vorhees, N.K. Gupta, and B. Johnson-Laird. The Collection Fusion Problem. In D. Harman, editor, *Proceedings of the* 3^{th} *Text REtrieval Conference*. NIST Special Publication 500-225, 1995.

Index of TREC-9 Papers by Task/Track

TREC home Publications home Help

Cross-Language

[*Filtering*] [*Interactive*] [*Query*] [*Question Answering*]
[*Spoken Document Retrieval*] [*Web*]

BBN Technologies

TREC-9 Cross Lingual Retrieval at BBN, *page 106*

Chinese University at Hong Kong

TREC-9 CLIR at CUHK Disambiguation by Similarity Values Between Adjacent Words, *page 151*

Fudan University

FDU at TREC-9: CLIR, Filtering and QA Tasks, *page 189*

IBM T.J. Watson Research Center

English-Chinese Informatin Retrieval at IBM, *page 223*

Johns Hopkins University, APL

The HAIRCUT System at TREC-9, *page 273*

Korea Advanced Institute of Science and Technology

TREC-9 Experiments at KAIST: QA, CLIR and Batch Filtering, *page 303*

Microsoft Research China, Tsinghua University China, and the Université de Montréal

TREC-9 CLIR Experiments at MSRCN, *page 343*

MNIS-TextWise Labs

CINDOR Trec-9 English-Chinese Evaluation, *page 379*

National Taiwan University

Description of NTU QA and CLIR Systems in TREC-9, *page 389*

Queens College, CUNY

TREC-9 Cross Language, Web and Question-Answering Track Experiments using PIRCS, *page 419*

RMIT University

Melbourne TREC-9 Experiments, *page 437*

University of California at Berkeley

TREC-9 Cross-Language Information Retrieval (English-Chinese) Overview, *page 15*

English-Chinese Cross-Language IR Using Bilingual Dictionaries, *page 517*

University of Maryland

TREC-9 Experiments at Maryland: Interactive CLIR, *page 543*

University of Massachusetts

INQUERY and TREC-9, *page 551*

Filtering

[*Cross-Language*] [*Interactive*] [*Query*] [*Question Answering*]
[*Spoken Document Retrieval*] [*Web*]

Carnegie Mellon University

kNN at TREC-9, *page 127*

YFilter at TREC-9, *page 135*

Fudan University

FDU at TREC-9: CLIR, Filtering and QA Tasks, *page 189*

Informatique-CDC, ESPCI

Training Context-Sensitive Neural Networks with Few Relevant Examples for the TREC-9 Routing, *page 257*

IRIT-SIG

Mercure at trec9: Web and Filtering tasks, *page 263*

KDD R&D Laboratories, Inc., Waseda University

Experiments on the TREC-9 Filtering Track, *page 295*

Korea Advanced Institute of Science and Technology

TREC-9 Experiments at KAIST: QA, CLIR and Batch Filtering, *page 303*

Microsoft Research, WhizzBang Labs

The TREC-9 Filtering Track Final Report, *page 25*

Microsoft Research Ltd., UK

Microsoft Cambridge at TREC-9: Filtering Track, *page 361*

Queens College, CUNY

TREC-9 Cross Language, Web and Question-Answering Track Experiments using PIRCS, *page 419*

Rutgers University

Logical Analysis of Data in the TREC-9 Filtering Track, *page 453*

Université de Montréal

The System RELIEFS: A New Approach for Information Filtering, *page 573*

University of Iowa

Filters and Answers: The University of Iowa TREC-9 Results, *page 533*

University of Nijmegen

Incrementality, Half-life, and Threshold Optimization for Adaptive Document Filtering, *page 589*

Interactive

[*Filtering*] [*Cross-Language*] [*Query*] [*Question Answering*]
[*Spoken Document Retrieval*] [*Web*]

Chapman University

Passive Feedback Collection--An Attempt to Debunk the Myth of Clickthroughs, *page 141*

National Institute of Standards and Technology and the Oregon Health Sciences University

TREC-9 Interactive Track Report, *page 41*

Oregon Health Sciences University

Further Analysis of Whether Batch and User Evaluations Give the Same Results with a Question-Answering Task, *page 407*

RMIT University

Melbourne TREC-9 Experiments, *page 437*

Rutgers University

Support for Question-Answering in Interactive Information Retrieval: Rutgers' TREC-9 Interactive Track Experience, *page 463*

University of Glasgow

Question Answering, Relevance Feedback and Summarisation: Trec-9 Interactive Track Report, *page 523*

University of Sheffield

Sheffield Interactive Experiment at TREC-9, *page 645*

Query

[*Cross-Language*] [*Filtering*] [*Interactive*]
[*Question Answering*] [*Spoken Document Retrieval*] [*Web*]

Hummingbird

Hummingbird's Fulcrum SearchServer at TREC-9, *page 211*

Microsoft Research Ltd., UK

Microsoft Cambridge at TREC-9: Filtering Track, *page 361*

National Institute of Standards and Technology

Query Expansion Seen Through Return Order of Relevant Documents, *page 51*

SabIR Research, Inc.

Query Expansion Seen Through Return Order of Relevant Documents, *page 51*

The TREC-9 Query Track, *page 81*

Sun Microsystems Laboratories

Halfway to Question Answering, *page 489*

University of Massachusetts

INQUERY and TREC-9, *page 551*

Question Answering

[*Cross-Language*] [*Filtering*] [*Interactive*]
[*Query*] [*Spoken Document Retrieval*] [*Web*]

CL Research

Syntactic Clues and Lexical Resources in Question-Answering, *page 157*

Fudan University

FDU at TREC-9: CLIR, Filtering and QA Tasks, *page 189*

IBM T.J. Watson Research Center

One Search Engine or Two for Question-Answering, *page 235*

IBM's Statistical Question Answering System, *page 231*

Imperial College of Science, Technology and Medicine

A Simple Question Answering System, *page 251*

Korea Advanced Institute of Science and Technology

TREC-9 Experiments at KAIST: QA, CLIR and Batch Filtering, *page 303*

Korea University

Question Answering Considering Semantic Categories and Co-Occurence Density, *page 317*

LIMSI-CNRS

QALC--The Question-Answering System of LIMSI-CNRS, *page 325*

Microsoft Research Ltd.

Question Answering Using a Large NLP System, *page 355*

The MITRE Corporation

Another Sys Called Qanda, *page 369*

National Institute of Standards and Technology

Overview of the TREC-9 Question Answering Track, *page 71*

National Taiwan University

Description of NTU QA and CLIR Systems in TREC-9, *page 389*

NTT Data Corporation

NTT DATA TREC-9 Question Answering Track Report, *page 399*

Queens College, CUNY

TREC-9 Cross Language, Web and Question-Answering Track Experiments using PIRCS, *page 419*

Southern Methodist University

FALCON: Boosting Knowledge for Answer Engines, *page 479*

Sun Microsystems Laboratories

Halfway to Question Answering, *page 489*

Syracuse University

Question Answering: CNLP at the TREC-9 Question Answering Track, *page 501*

Universidad de Alicante

A Semantic Approach to Question Answering Systems, *page 511*

Universitá di Pisa - Italy

The PISAB Question Answering System, *page 621*

Université de Montréal

Goal-Driven Answer Extraction, *page 563*

University of Iowa

Filters and Answers: The University of Iowa TREC-9 Results, *page 533*

University of Massachusetts

INQUERY and TREC-9, *page 551*

University of Sheffield

University of Sheffield TREC-9 QA System, *page 635*

University of Southern California

Question Answering in Webclopedia, *page 655*

University of Waterloo, CTIT

Question Answering by Passage Selection (MultiText Experiments for TREC-9), *page 673*

Spoken Document Retrieval

[*Cross-Language*] [*Filtering*] [*Interactive*] [*Query*]
[*Question Answering*] [*Web*]

Cambridge University

Spoken Document Retrieval for TREC-9 at Cambridge University, *page 117*

LIMSI

The LIMSI SDR System for TREC-9, *page 335*

National Institute of Standards and Technology

Spoken Document Retrieval Track Slides

University of Sheffield

The Thisl SDR System at TREC-9, *page 627*

Web

[*Cross-Language*] [*Filtering*] [*Interactive*] [*Query*]
[*Question Answering*] [*Spoken Document Retrieval*]

AT&T Labs-Research

AT&T Labs at TREC-9, *page 103*

CSIRO Mathematics and Information Sciences

ACSys/CSIRO TREC-9 Experiments, *page 167*

Melbourne TREC-9 Experiments, *page 437*

Overview of the TREC-9 Web Track, *page 87*

CWI, Amsterdam

The Mirror DBMS at TREC-9, *page 171*

Dublin City University

Dublin City University Experiments in Connectivity Analysis for TREC-9, *page 179*

Fujitsu Laboratories, Ltd.

Fujitsu Laboratories TREC-9 Report, *page 203*

Hummingbird

Hummingbird's Fulcrum SearchServer at TREC-9, *page 211*

Illinois Institute of Technology

IIT TREC-9-Entity Based Feedback with Fusion, *page 241*

IRIT-SIG

Mercure at trec9: Web and Filtering tasks, *page 263*

Johns Hopkins University, APL

The HAIRCUT System at TREC-9, *page 273*

Justsystem Corporation

Reflections on "Aboutness" TREC-9 Evaluation Experiments at Justsystem, *page 281*

Queens College, CUNY

TREC-9 Cross Language, Web and Question-Answering Track Experiments using PIRCS, *page 419*

RICOH Co., Ltd.

Structuring and Expanding Queries in the Probablistic Model, *page 427*

RMIT University

Melbourne TREC-9 Experiments, *page 437*

SabIR Research, Inc.

SabIR Research at TREC-9, *page 475*

TNO-TPD and Univ. of Twente

TNO-UT at TREC-9: How Different are Web Documents?, *page 665*

University of Bangkok, Thailand

Kasetsart University TREC-9 Experiments, *page 289*

Université de Neuchâtel

Report on the TREC-9 Experiment: Link-based Retrieval an Distributed Collections, *page 579*

University of North Carolina, Chapel Hill

Information Space Based on HTML Structure, *page 601*

University of Padova, Italy

Web Document Retrieval Using Passage Retrieval, Connectivity Information, and Automatic Link Weighting--TREC-9 Report, *page 611*

University of Waterloo, CTIT

Question Answering by Passage Selection (MultiText Experiments for TREC-9), *page 673*

Last updated: Friday, 05-Oct-01 08:17:36
Date created: Wednesday, 01-Aug-17
trec@nist.gov

CROSS-LANGUAGE TRACK

Tag	Organization	Monolingual	Run Type
BBN9XLA	BBN Technologies	Cross	Automatic, T+D+N
BBN9XLB	BBN Technologies	Cross	Automatic, T+D+N
BBN9XLC	BBN Technologies	Cross	Automatic, T+D+N
BBN9MONO	BBN Technologies	Mono	Automatic, T+D+N
CHUHK00CH1	Chinese Univ. of Hong Kong	Mono	Automatic, T+D+N
CHUHK00XEC1	Chinese Univ. of Hong Kong	Cross	Automatic, T+D+N
Fdut9xl1	Fudan Univ.	Cross	Automatic, T+D
Fdut9xl2	Fudan Univ.	Cross	Automatic, T+D
Fdut9xl3	Fudan Univ.	Cross	Automatic, D
Fdut9xl4	Fudan Univ.	Mono	Automatic, T+D
Ibmcl9a	IBM T.J. Watson (Yorktown Heights)	Cross	Automatic, T+D+N
Ibmcl9s	IBM T.J. Watson (Yorktown Heights)	Cross	Automatic, T+D+N
Ibmcl9m	IBM T.J. Watson (Yorktown Heights)	Mono	Automatic, T+D+N
Apl9xcmb	Johns Hopkins Univ., APL	Cross	Automatic, T+D+N
Apl9xtop	Johns Hopkins Univ., APL	Cross	Automatic, T+D+N
Apl9xwrd	Johns Hopkins Univ., APL	Cross	Automatic, T+D+N
Apl9xmon	Johns Hopkins Univ., APL	Mono	Automatic, T+D+N
KAIST9xlmt	KAIST	Cross	Automatic, T+D
KAIST9xlqm	KAIST	Cross	Automatic, T+D
KAIST9xlqt	KAIST	Cross	Automatic, T+D
KAIST9xlch	KAIST	Mono	Automatic, T+D
msrcn1	Microsoft Research, China	Cross	Automatic, T+D+N
msrcn2	Microsoft Research, China	Cross	Automatic, T+D+N
msrcn3	Microsoft Research, China	Mono	Automatic, T+D+N
TWe2c3Cltdn	MNIS-TextWise Labs	Cross	Automatic, T+D+N
TWmono3Cltdn	MNIS-TextWise Labs	Mono	Automatic, T+D+N
Ecirntua1	National Taiwan Univ.	Cross	Automatic, T+D
Ecirntuco	National Taiwan Univ.	Cross	Automatic, T+D
Pir0XHxD	Queens College, CUNY	Cross	Automatic, T+D+N
Pir0Xdin	Queens College, CUNY	Cross	Automatic, T+D+N
Pir0Xhnd	Queens College, CUNY	Cross	Automatic, T+D+N
Pir0Xori	Queens College, CUNY	Mono	Automatic, T+D
Rmitcl001	RMIT Univ./CSIRO	Cross	Automatic, T+D
Rmitcl002	RMIT Univ./CSIRO	Cross	Automatic, T+D
Rmitcl003	RMIT Univ./CSIRO	Cross	Automatic, T+D
Rmitcl004	RMIT Univ./CSIRO	Mono	Automatic, T+D
Transezbig1	Trans-EZ Inc.	Cross	Automatic, T+D
Transezbig2	Trans-EZ Inc.	Cross	Automatic, T+D
Transezbig3	Trans-EZ Inc.	Cross	Automatic, T+D
Transezmono	Trans-EZ Inc.	Mono	Automatic, T+D
BRKECA1	Univ. of California, Berkeley	Cross	Automatic, T+D+N
BRKECA2	Univ. of California, Berkeley	Cross	Automatic, T+D+N
BRKECM1	Univ. of California, Berkeley	Cross	Manual
BRKCCA1	Univ. of California, Berkeley	Mono	Automatic, T+D+N
TB	Univ. of Maryland	Cross	Automatic, T+D+N
mixed	Univ. of Maryland	Cross	Automatic, T+D+N
Percent	Univ. of Maryland	Cross	Automatic, T+D+N
INQ7XL1	Univ. of Massachusetts	Cross	Automatic, T+D
INQ7XL3	Univ. of Massachusetts	Cross	Automatic, T+D
INQ7XL4	Univ. of Massachusetts	Cross	Automatic, T+D
INQ7XL2	Univ. of Massachusetts	Mono	Automatic, T+D

BEST COPY AVAILABLE

TREC-9 Filtering Results

Publications home NIST Special Publication XXX- XXX

Help

- Carnegie Mellon University (DIR) - adaptive
- Carnegie Mellon University (CAT) - adaptive
- Carnegie Mellon University (CAT) - batch
- Fudan University - adaptive
- Fudan University - batch
- Informatique-CDC, DTA - routing
- Informatique-CDC, DTA - batch
- IRIT/SIG - batch
- IRIT/SIG - routing
- KAIST (Korea Advanced Institute of Science and Technology) - batch
- Katholieke Universiteit Nijmegen - adaptive
- /Katholieke Universiteit Nijmegen - batch
- Katholieke Universiteit Nijmegen - routing
- KDD Labs/Waseda University - adaptive
- Microsoft Research Ltd - adaptive
- Microsoft Research Ltd - batch
- Microsoft Research Ltd - routing
- Queens College, CUNY - adaptive
- Rutgers University (Kantor) - adaptive
- Rutgers University (Kantor) - routing
- Seoul National University - batch

- University of Iowa - adaptive
- University of Montreal - adaptive

Last updated: Wednesday, 29-Aug-01 10:02:47
Date created: Tuesday, 28-Aug-01
trec@nist.gov

TREC-9 Interactive Results

Publications home NIST Special Publication XXX- XXX

Help NIST HOME

- Chapman University
- CSIRO/RMIT
- Glasgow University
- Oregon Health Sciences University
- Rutgers University
- Sheffield University

Last updated: Wednesday, 29-Aug-01 15:58:16
Date created: Tuesday, 28-Aug-01
trec@nist.gov

Run CMUDIR11	
Sub task	adaptive
Query construction	automatic
.M field used?	no
TREC data used in training?	no
non-TREC data used ?	no
Optimized for	T9P
Topic set	MESH
Total retrieved	333532
Relevant retrieved	142548
Macro average recall	0.185
Macro average precision	0.359
Mean T9P	0.359
Mean Utility	19.191
Mean T9U	19.191
Mean scaled utility	-0.001
Zero returns	0

Run CMUDIR11	
Sub task	adaptive
Query construction	automatic
.M field used?	no
TREC data used in training?	no
non-TREC data used ?	no
Optimized for	T9P
Topic set	MESH-SAMPLE
Total retrieved	33392
Relevant retrieved	14043
Macro average recall	0.189
Macro average precision	0.363
Mean T9P	0.363
Mean Utility	17.474
Mean T9U	17.474
Mean scaled utility	0.005
Zero returns	0

Run CMUDIR12	
Sub task	adaptive
Query construction	automatic
.M field used?	no
TREC data used in training?	no
non-TREC data used ?	no
Optimized for	T9P
Topic set	MESH-SAMPLE
Total retrieved	33392
Relevant retrieved	14043
Macro average recall	0.189
Macro average precision	0.363
Mean T9P	0.363
Mean Utility	17.474
Mean T9U	17.474
Mean scaled utility	0.005
Zero returns	0

Run CMUDIR13	
Sub task	adaptive
Query construction	automatic
.M field used?	no
TREC data used in training?	no
non-TREC data used ?	no
Optimized for	T9U
Topic set	MESH-SAMPLE
Total retrieved	15753
Relevant retrieved	9708
Macro average recall	0.137
Macro average precision	0.471
Mean T9P	0.306
Mean Utility	26.742
Mean T9U	26.742
Mean scaled utility	0.084
Zero returns	0

Run CMUDIR14	
Sub task	adaptive
Query construction	automatic
.M field used?	no
TREC data used in training?	no
non-TREC data used ?	no
Optimized for	T9P
Topic set	OHSU
Total retrieved	3798
Relevant retrieved	1102
Macro average recall	0.363
Macro average precision	0.267
Mean T9P	0.267
Mean Utility	-7.810
Mean T9U	-7.810
Mean scaled utility	-0.647
Zero returns	0

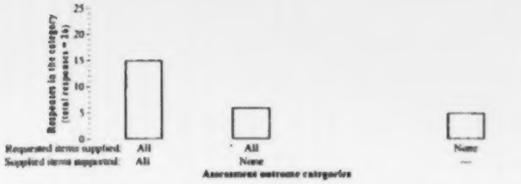
Run CMUDIR15	
Sub task	adaptive
Query construction	automatic
.M field used?	no
TREC data used in training?	no
non-TREC data used ?	no
Optimized for	T9U
Topic set	OHSU
Total retrieved	1093
Relevant retrieved	559
Macro average recall	0.170
Macro average precision	0.378
Mean T9P	0.167
Mean Utility	9.270
Mean T9U	9.270
Mean scaled utility	0.008
Zero returns	0

Run CMUDIR16	
Sub task	adaptive
Query construction	automatic
.M field used?	yes
TREC data used in training?	no
non-TREC data used ?	no
Optimized for	T9P
Topic set	OHSU
Total retrieved	3842
Relevant retrieved	1169
Macro average recall	0.414
Macro average precision	0.279
Mean T9P	0.279
Mean Utility	-5.317
Mean T9U	-5.317
Mean scaled utility	-0.575
Zero returns	0

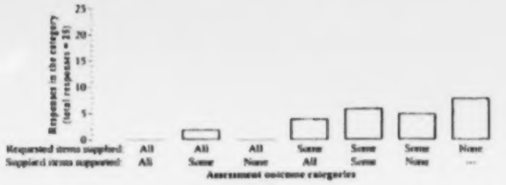
Run CMUDIR17	
Sub task	adaptive
Query construction	automatic
.M field used?	yes
TREC data used in training?	no
non-TREC data used ?	no
Optimized for	T9U
Topic set	OHSU
Total retrieved	1062
Relevant retrieved	566
Macro average recall	0.190
Macro average precision	0.426
Mean T9P	0.166
Mean Utility	10.095
Mean T9U	10.095
Mean scaled utility	0.042
Zero returns	0

Run CMUDIR18	
Sub task	adaptive
Query construction	automatic
.M field used?	no
TREC data used in training?	no
non-TREC data used ?	no
Optimized for	T9U
Topic set	MESH
Total retrieved	97722
Relevant retrieved	65833
Macro average recall	0.098
Macro average precision	0.479
Mean T9P	0.233
Mean Utility	20.346
Mean T9U	20.346
Mean scaled utility	0.067
Zero returns	0

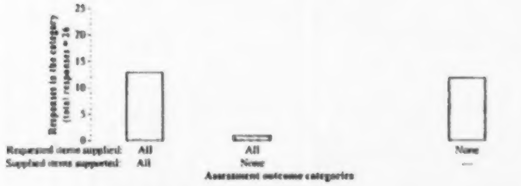
Run CMUDIR18	
Sub task	adaptive
Query construction	automatic
.M field used?	no
TREC data used in training?	no
non-TREC data used ?	no
Optimized for	T9U
Topic set	MESH-SAMPLE
Total retrieved	10066
Relevant retrieved	6847
Macro average recall	0.107
Macro average precision	0.507
Mean T9P	0.239
Mean Utility	20.950
Mean T9U	20.950
Mean scaled utility	0.077
Zero returns	0



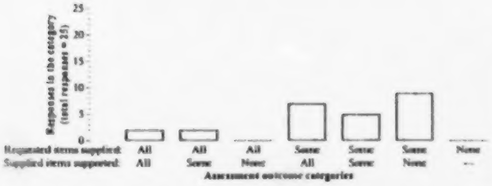
7. Which was the last dynasty of China: Qing or Ming?



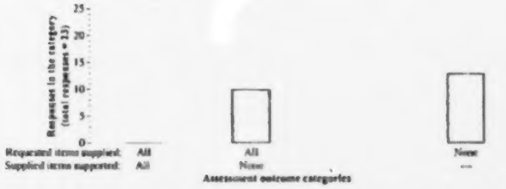
4. Name three countries that imported Cuban sugar....



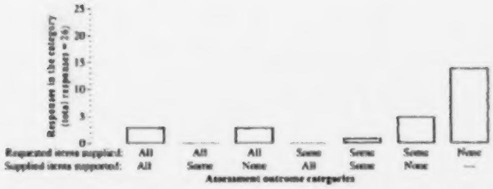
5. Which children's TV program was on the air longer: the original Mickey Mouse Club or the original Howdy Doody Show?



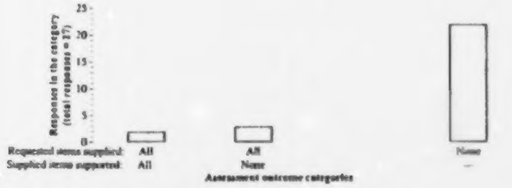
3. Name four films in which Orson Welles appeared.



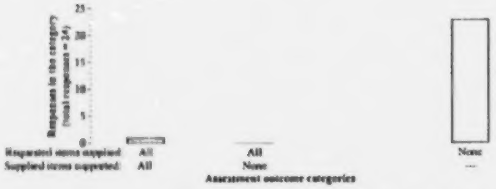
6. Which painting did Edvard Munch complete first: "Vampire" or "Puberty"?



1. What are the names of 3 US national parks where one can find redwoods?

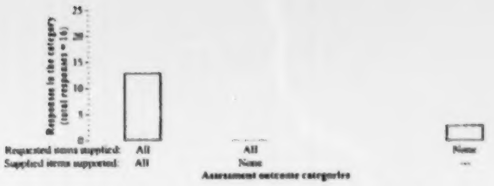


8. Is Denmark larger or smaller in population than Norway?

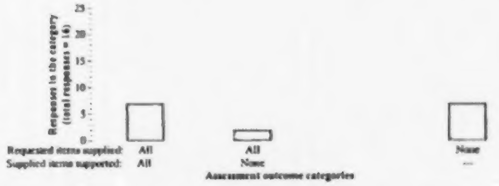


2. Identify a site with Roman ruins in present day France.

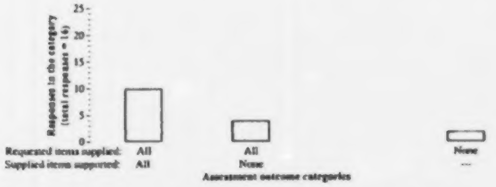
BEST COPY AVAILABLE



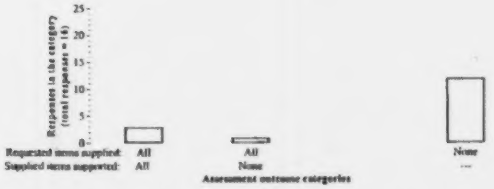
7. Which was the last dynasty of China: Qing or Ming?



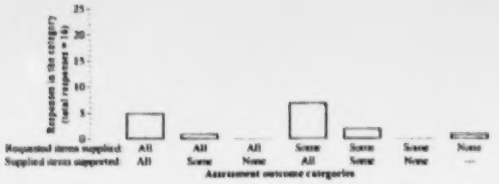
5. Which children's TV program was on the air longer: the original Mickey Mouse Club or the original Howdy Doody Show?



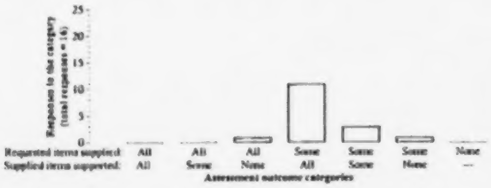
6. Which painting did Edvard Munch complete first: "Vampire" or "Puberty"?



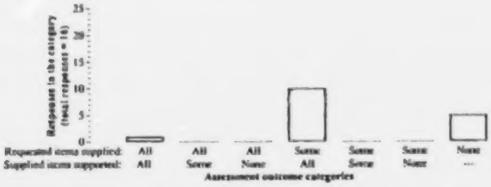
8. Is Denmark larger or smaller in population than Norway?



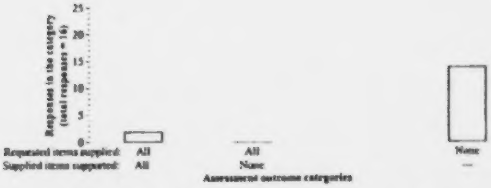
4. Name three countries that imported Cuban sugar....



3. Name four films in which Orson Welles appeared.

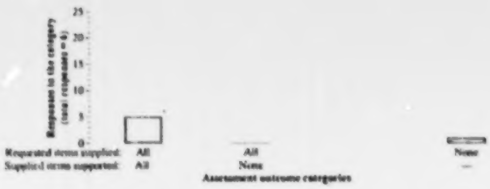


1. What are the names of 3 US national parks where one can find redwoods?

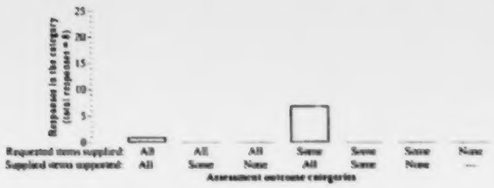


2. Identify a site with Roman ruins in present day France.

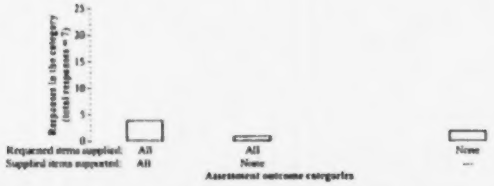
BEST COPY AVAILABLE



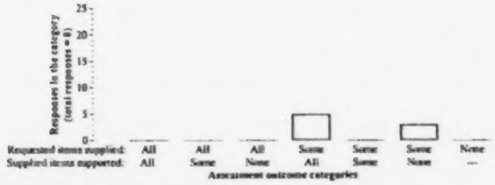
7. Which was the last dynasty of China: Qing or Ming?



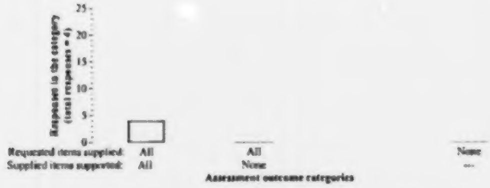
4. Name three countries that imported Cuban sugar....



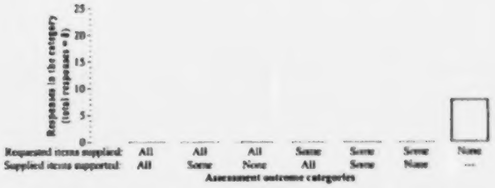
5. Which children's TV program was on the air longer: the original Mickey Mouse Club or the original Howdy Doody Show?



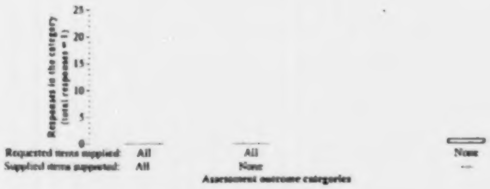
3. Name four films in which Orson Welles appeared.



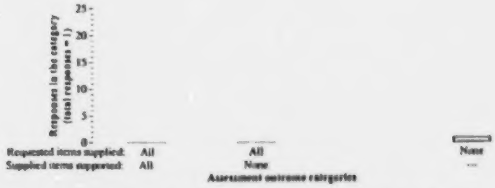
6. Which painting did Edvard Munch complete first: "Vampire" or "Puberty"?



1. What are the names of 3 US national parks where one can find redwoods?

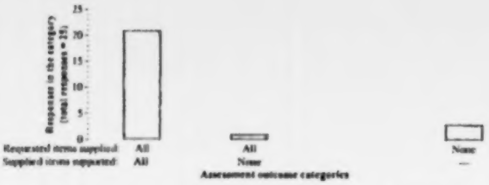


8. Is Denmark larger or smaller in population than Norway?

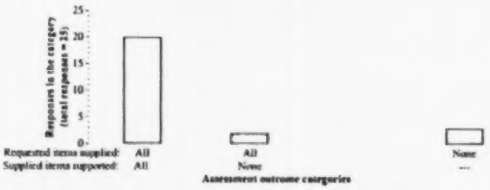


2. Identify a site with Roman ruins in present day France.

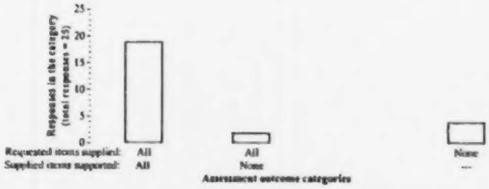
BEST COPY AVAILABLE



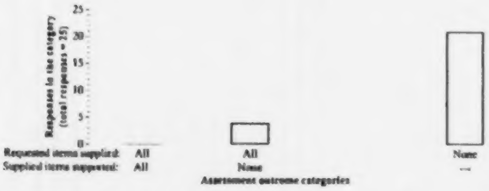
7. Which was the last dynasty of China: Qing or Ming?



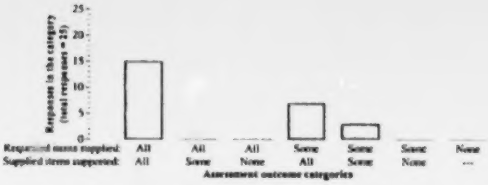
5. Which children's TV program was on the air longer: the original Mickey Mouse Club or the original Howdy Doody Show?



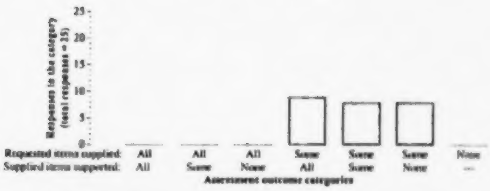
6. Which painting did Edvard Munch complete first: "Vampire" or "Puberty"?



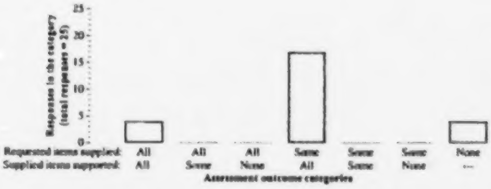
8. Is Denmark larger or smaller in population than Norway?



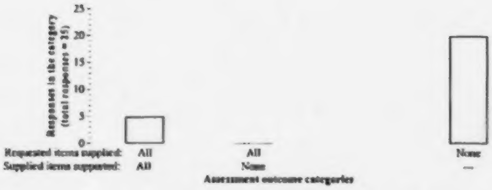
4. Name three countries that imported Cuban sugar....



3. Name four films in which Orson Welles appeared.

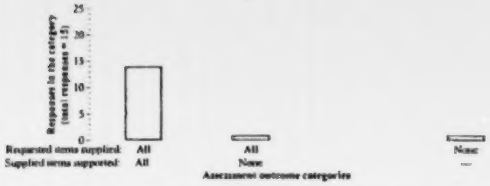


1. What are the names of 3 US national parks where one can find redwoods?

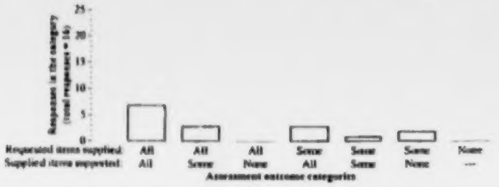


2. Identify a site with Roman ruins in present day France.

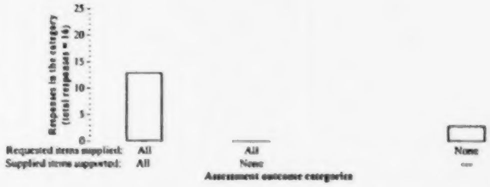
BEST COPY AVAILABLE



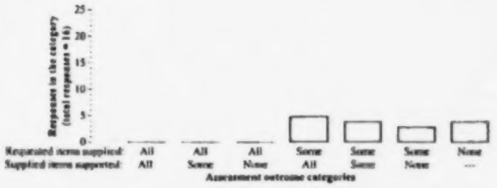
7. Which was the last dynasty of China: Qing or Ming?



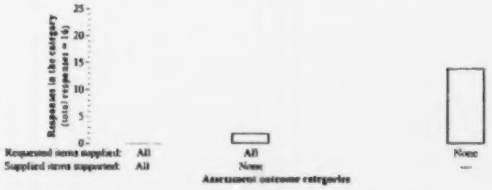
4. Name three countries that imported Cuban sugar....



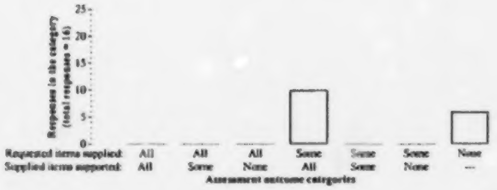
5. Which children's TV program was on the air longer: the original Mickey Mouse Club or the original Howdy Doody Show?



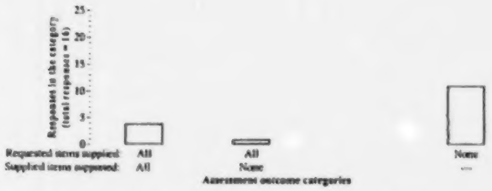
3. Name four films in which Orson Welles appeared.



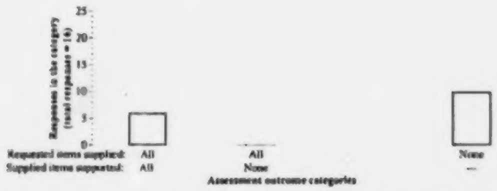
6. Which painting did Edvard Munch complete first: "Vampire" or "Puberty"?



1. What are the names of 3 US national parks where one can find redwoods?

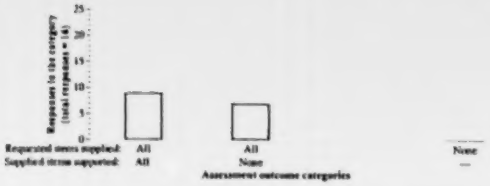


8. Is Denmark larger or smaller in population than Norway?

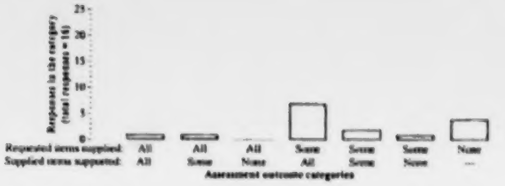


2. Identify a site with Roman ruins in present day France.

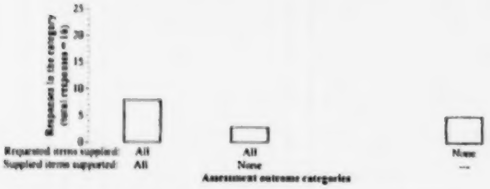
BEST COPY AVAILABLE



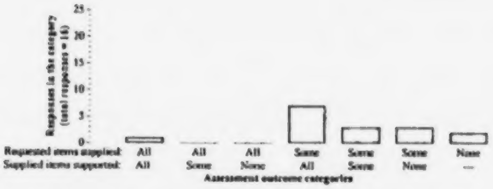
7. Which was the last dynasty of China: Qing or Ming?



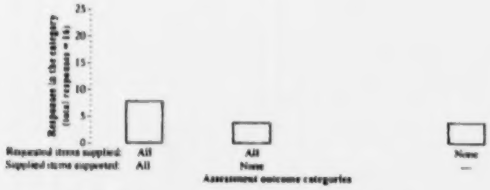
4. Name three countries that imported Cuban sugar....



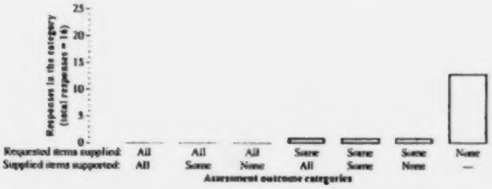
5. Which children's TV program was on the air longer: the original Mickey Mouse Club or the original Howdy Doody Show?



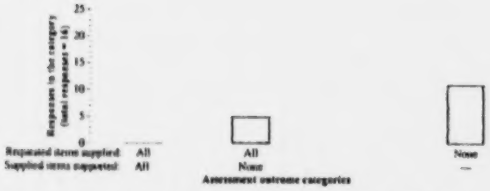
3. Name four films in which Orson Welles appeared.



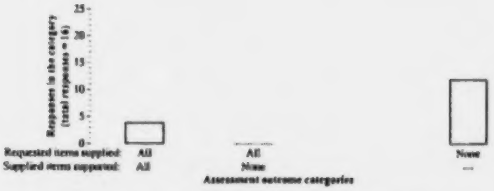
6. Which painting did Edvard Munch complete first: "Vampire" or "Puberty"?



1. What are the names of 3 US national parks where one can find redwoods?



8. Is Denmark larger or smaller in population than Norway?



2. Identify a site with Roman ruins in present day France.

BEST COPY AVAILABLE

TREC-9 Question Answering Results

Publications home NIST Special Publication XXX-XXX

Help NIST HOME

- MTR00S1
- MTR00L1
- INQ9WSUM
- INQ9AND
- msq9L50
- msq9L250
- ibmhlt0050
- ibmhlt00250
- shef50ea
- shef50
- shef250
- shef250p
- pir0qas1
- pir0qas2
- pir0qal1
- pir0qal2
- ICrjc99a
- ICrjc99b
- clr00b1

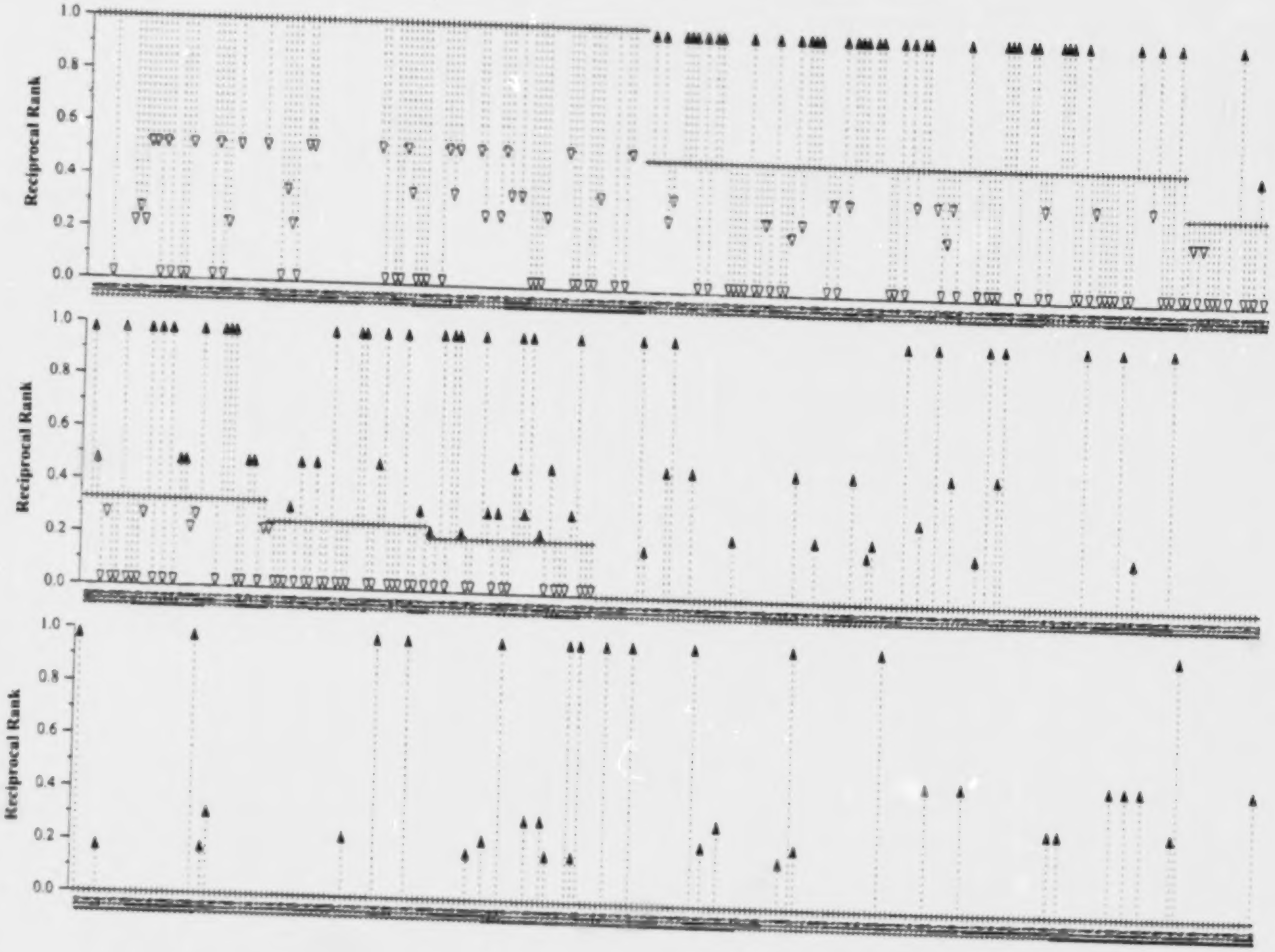
- clr00b2
- clr00s1
- clr00s2
- KUQA250a
- LCCSMU1
- LCCSMU2
- ALI9A250
- ALI9A50
- ALI9C250
- ALI9C50
- NTTD9QAa1S
- NTTD9QAa2S
- KAIST9qa1
- KAIST9qa2
- NTTD9QAa1L
- NTTD9QAb1L
- lcat050
- lcat250
- lcix250
- FDUT9QS1
- FDUT9QL1
- UdeMlng1
- UdeMlng2
- UdeMshrt
- UdeMexct

- PISA0
- IBMKR250
- IBMKA250
- IBMKA50
- IBMKR50
- FDUT9QL2
- PISAB
- xeroxQA9s
- xeroxQA9l
- PISAS
- FDUT9QS2
- SUT9p2c3c050
- SUT9p2c3c250
- SUT9bn3c050
- SUT9bn3c250
- uwmt9qas0
- uwmt9qal0
- qntua02
- qntua01
- qntua03
- cnxrole2
- Scai9QnA2
- Scai9QnA3
- ualberta
- ISI0A50

- SunOne
- SunToo
- uwmt9qas1
- uwmt9qal1
- UIQA002
- UIQA001

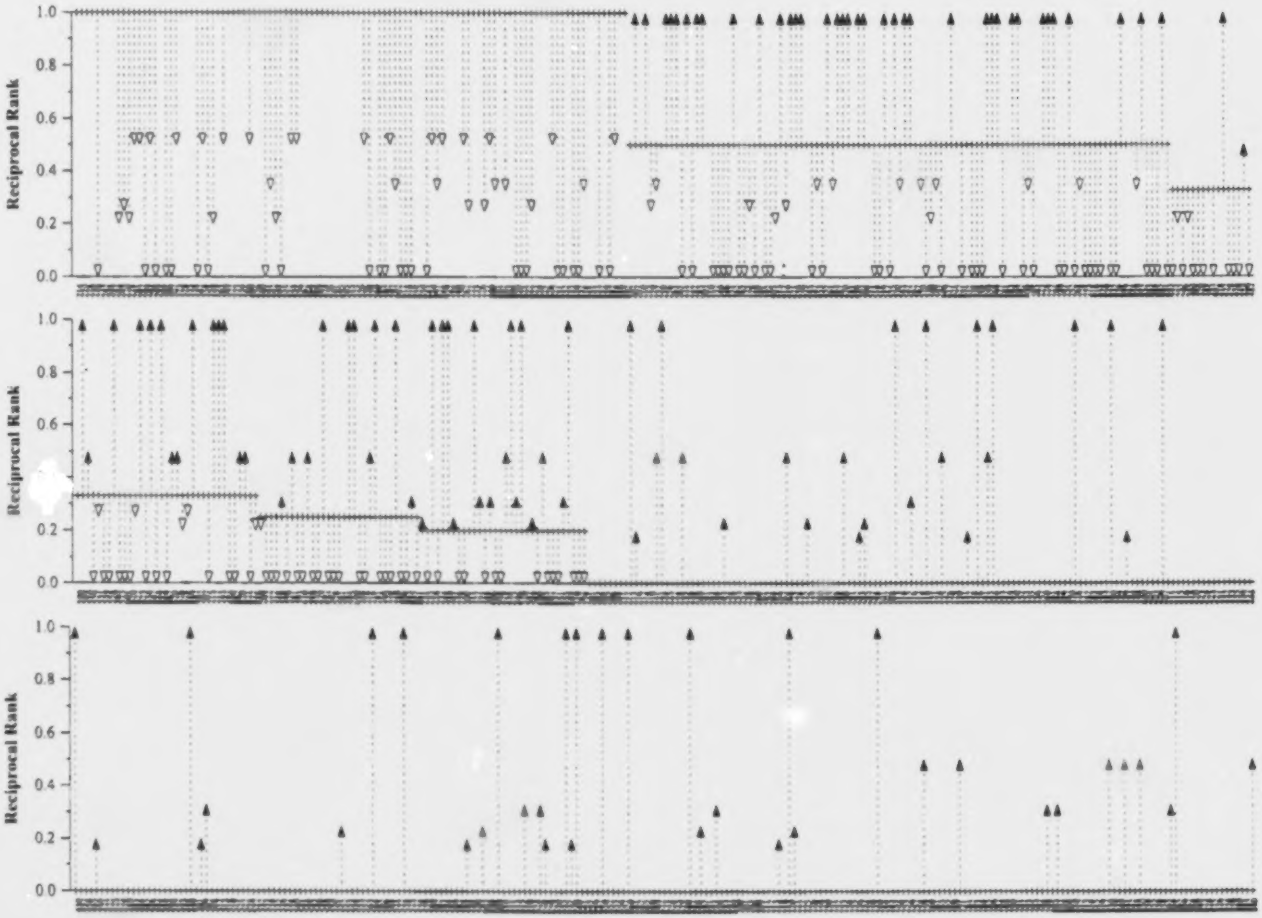
Last updated: Tuesday, 28-Aug-01 09:46:24
Date created: Tuesday, 28-Aug-01
trec@nist.gov

Run:	msq9L250
Category :	250 byte limit
Mean reciprocal rank:	0.264
# questions with no answer found:	417
Average answer length:	68.080 bytes

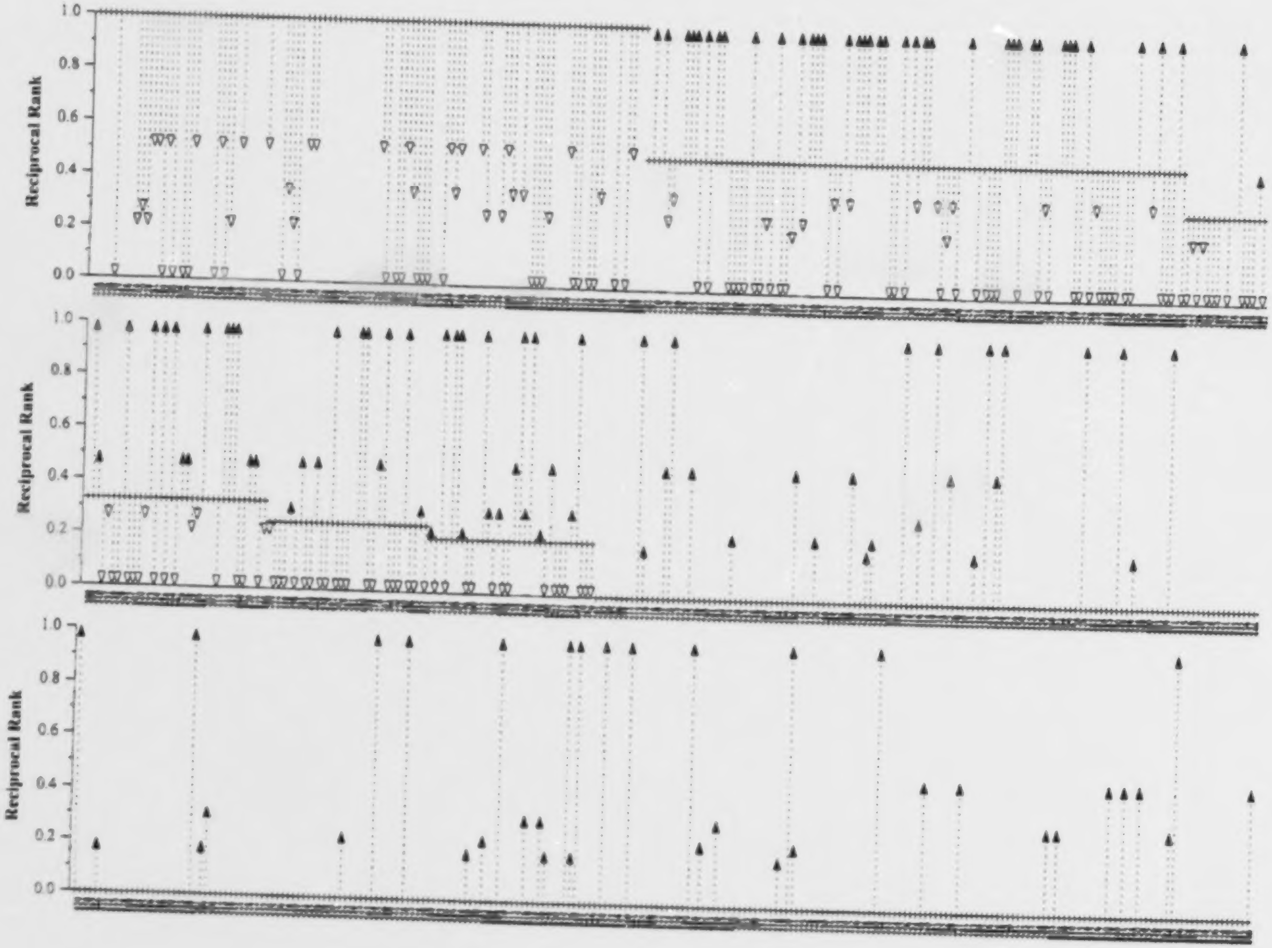


BEST COPY AVAILABLE

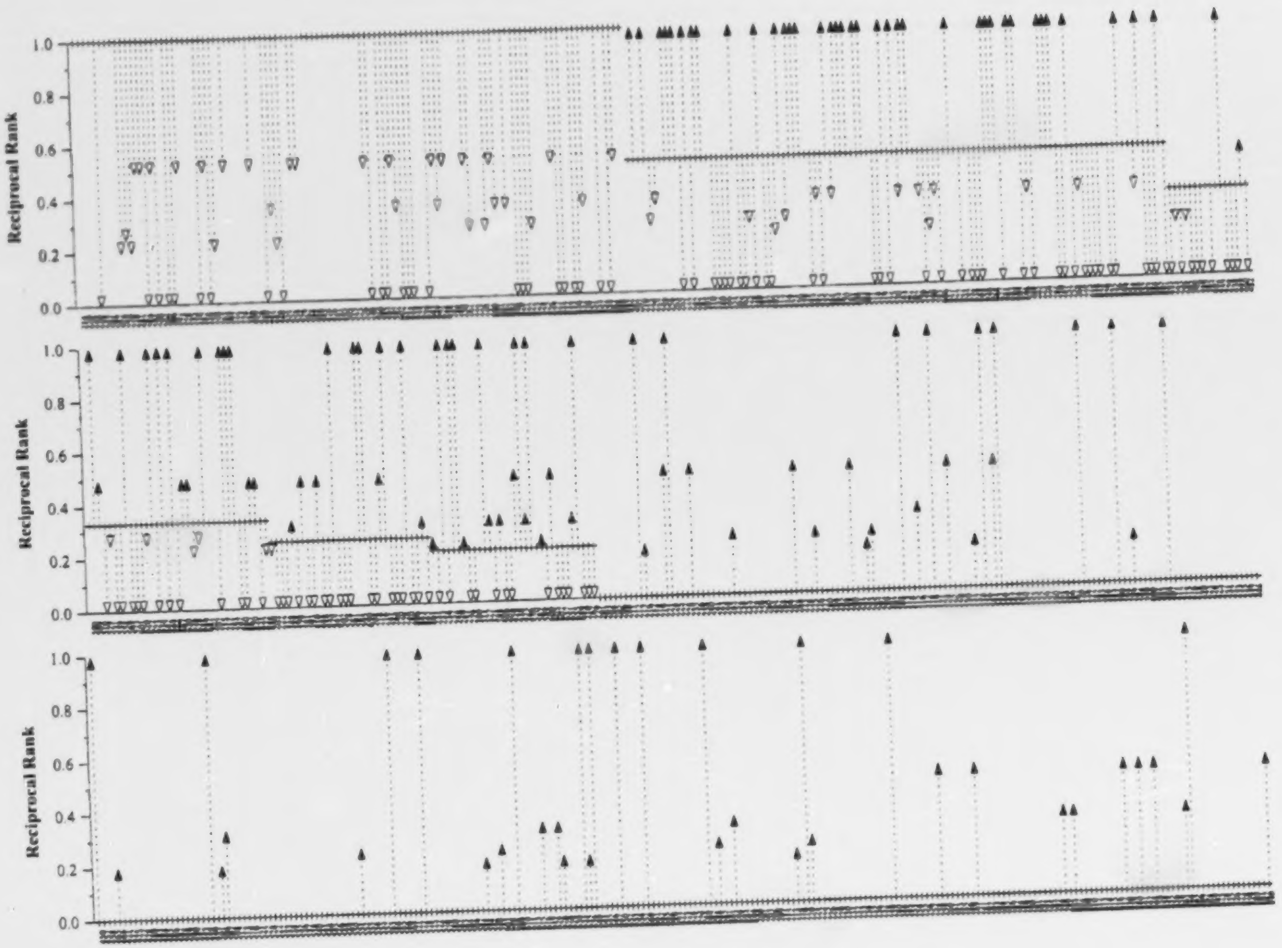
Run:	msq9L250
Category :	250 byte limit
Mean reciprocal rank:	0.264
# questions with no answer found:	417
Average answer length:	68.080 bytes



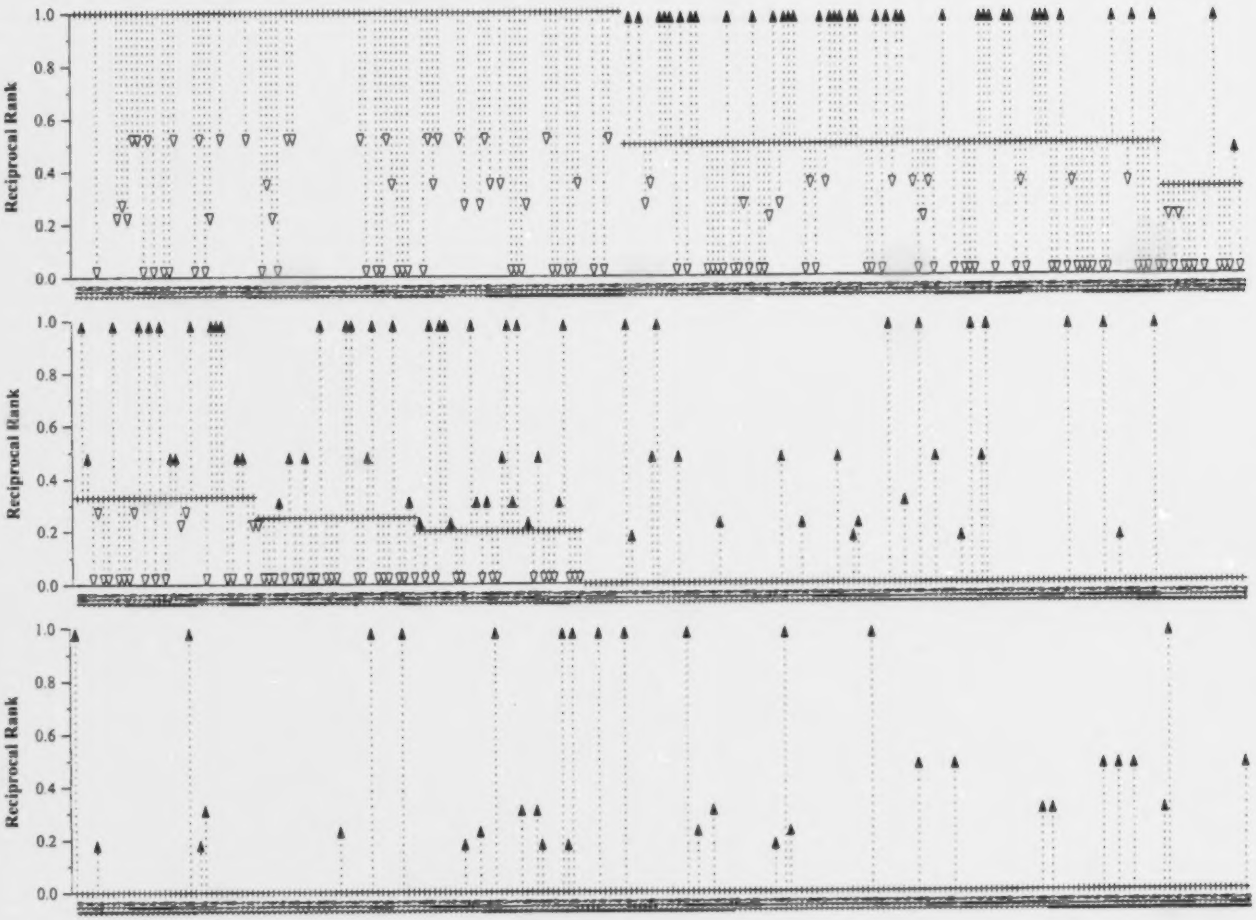
Run:	msq9L250
Category :	250 byte limit
Mean reciprocal rank:	0.264
# questions with no answer found:	417
Average answer length:	68.080 bytes



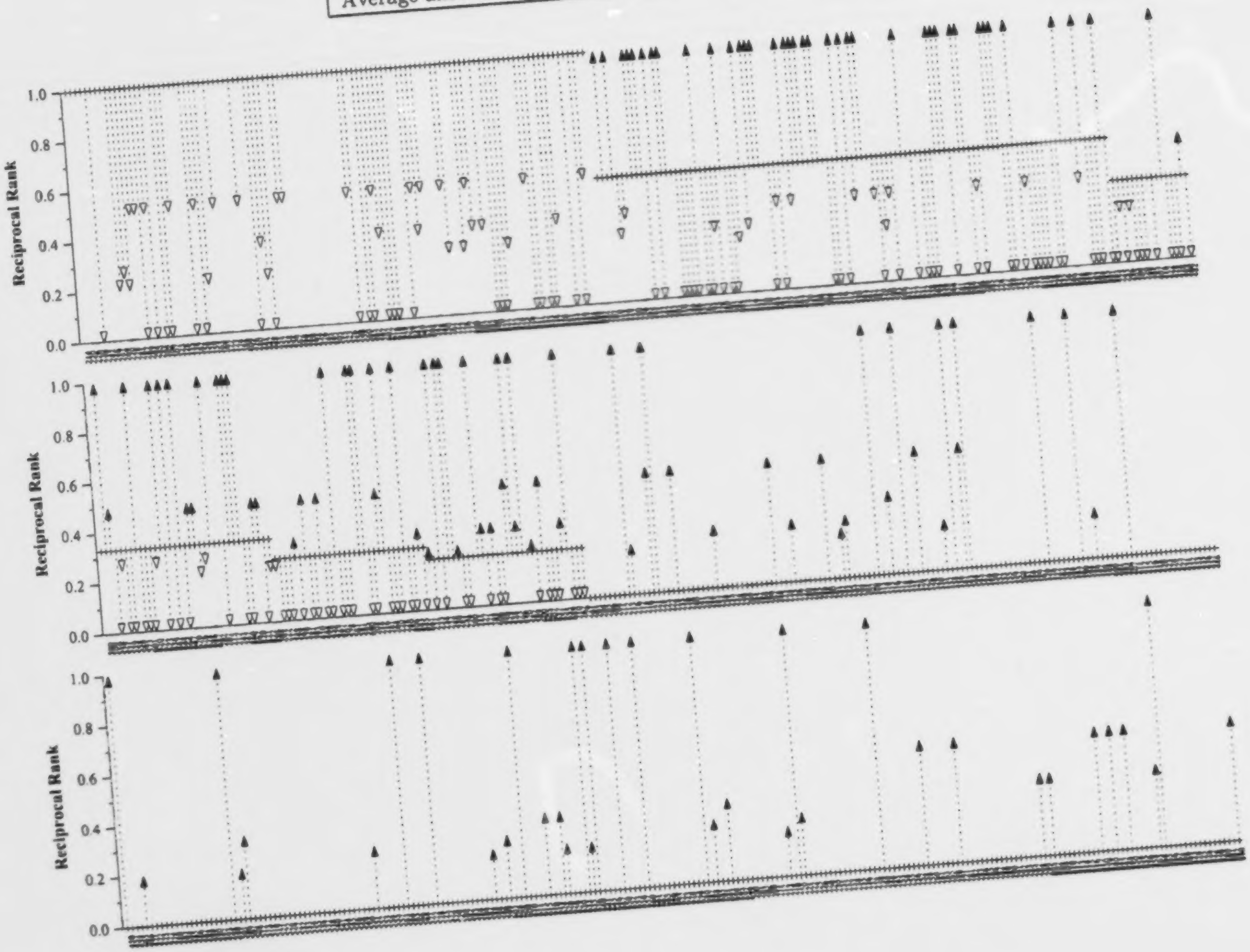
Run:	msq9L250
Category :	250 byte limit
Mean reciprocal rank:	0.264
# questions with no answer found:	417
Average answer length:	68.080 bytes



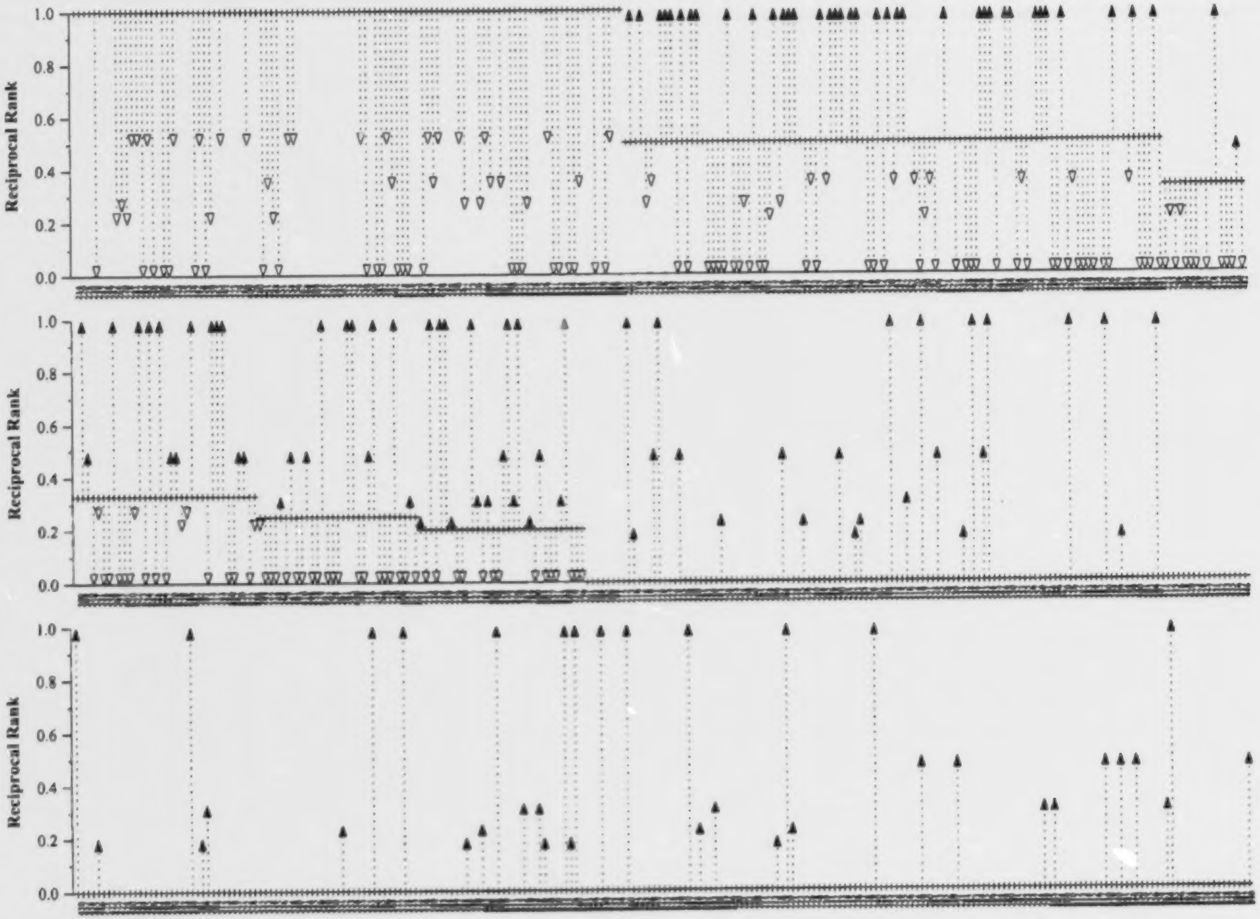
Run:	msq9L250
Category :	250 byte limit
Mean reciprocal rank:	0.264
# questions with no answer found:	417
Average answer length:	68.080 bytes



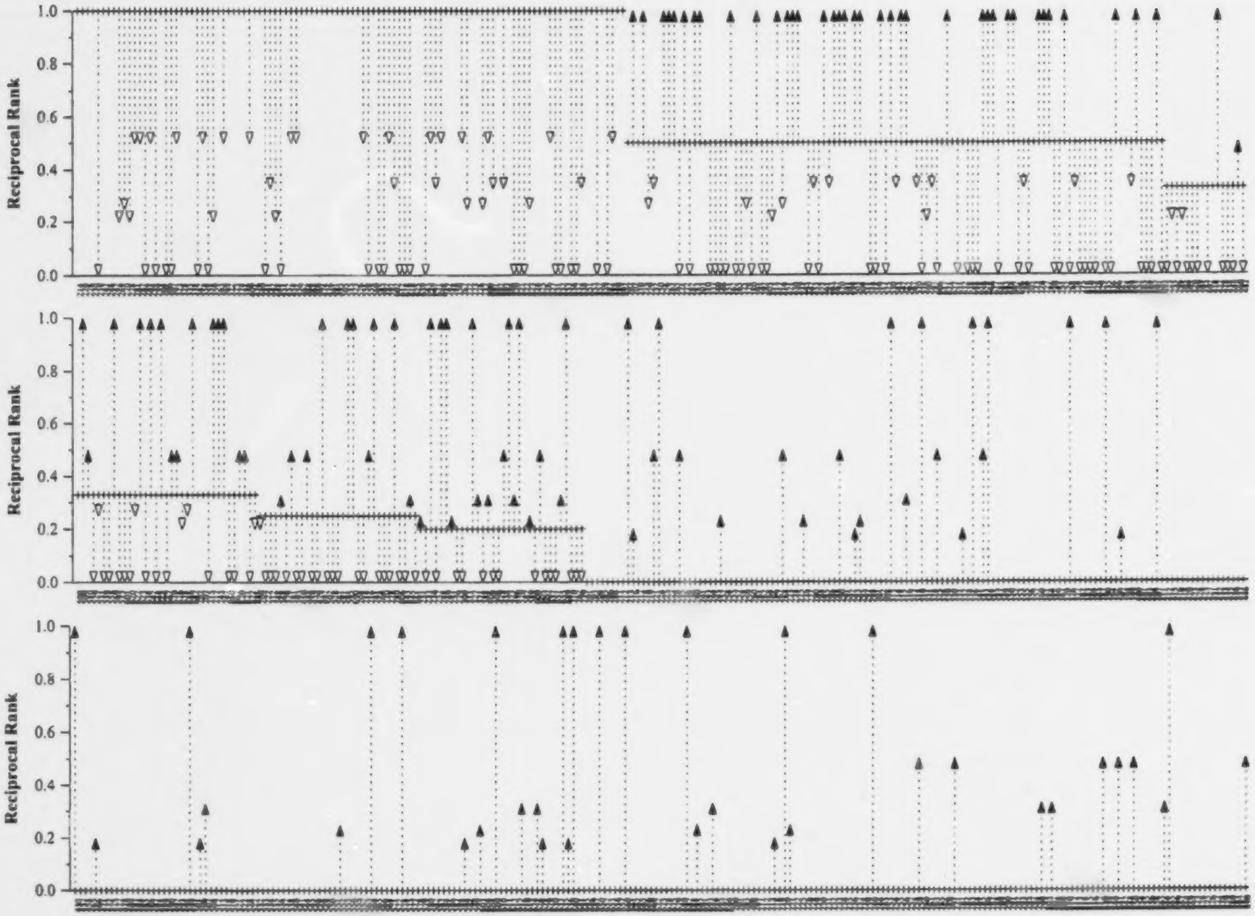
Run:	msq9L250
Category :	250 byte limit
Mean reciprocal rank:	0.264
# questions with no answer found:	417
Average answer length:	68.080 bytes



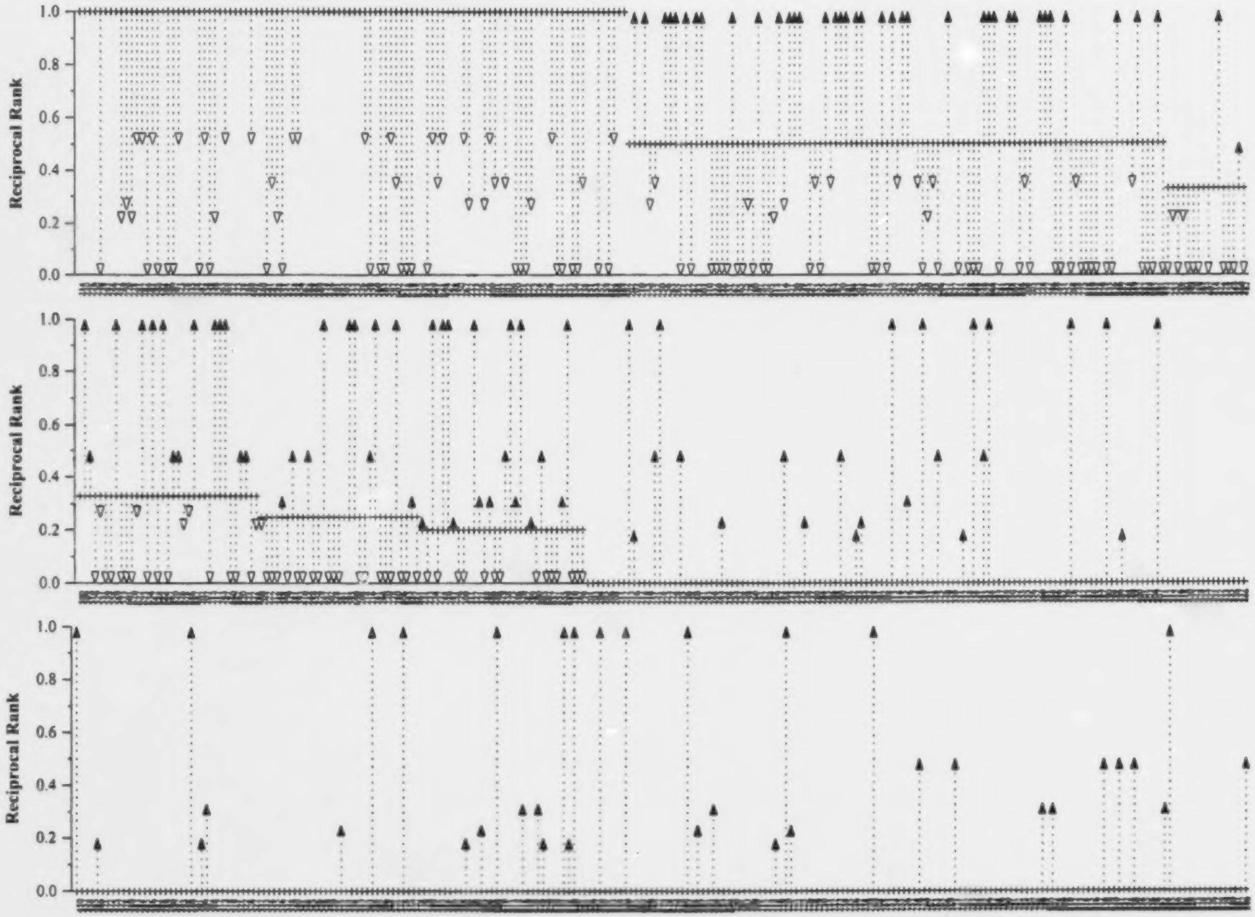
Run:	msq9L250
Category :	250 byte limit
Mean reciprocal rank:	0.264
# questions with no answer found:	417
Average answer length:	68.080 bytes



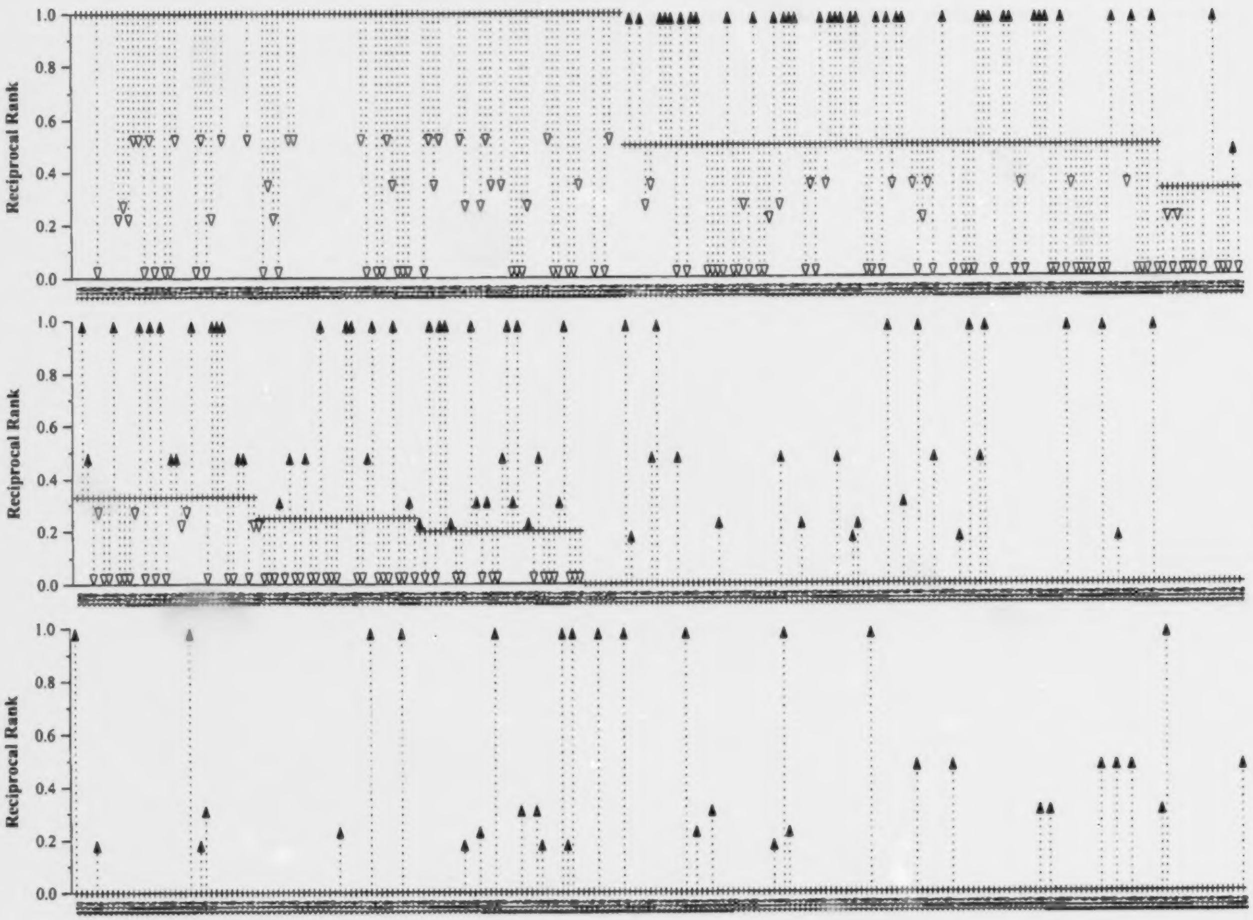
Run:	msq9L250
Category :	250 byte limit
Mean reciprocal rank:	0.264
# questions with no answer found:	417
Average answer length:	68.080 bytes



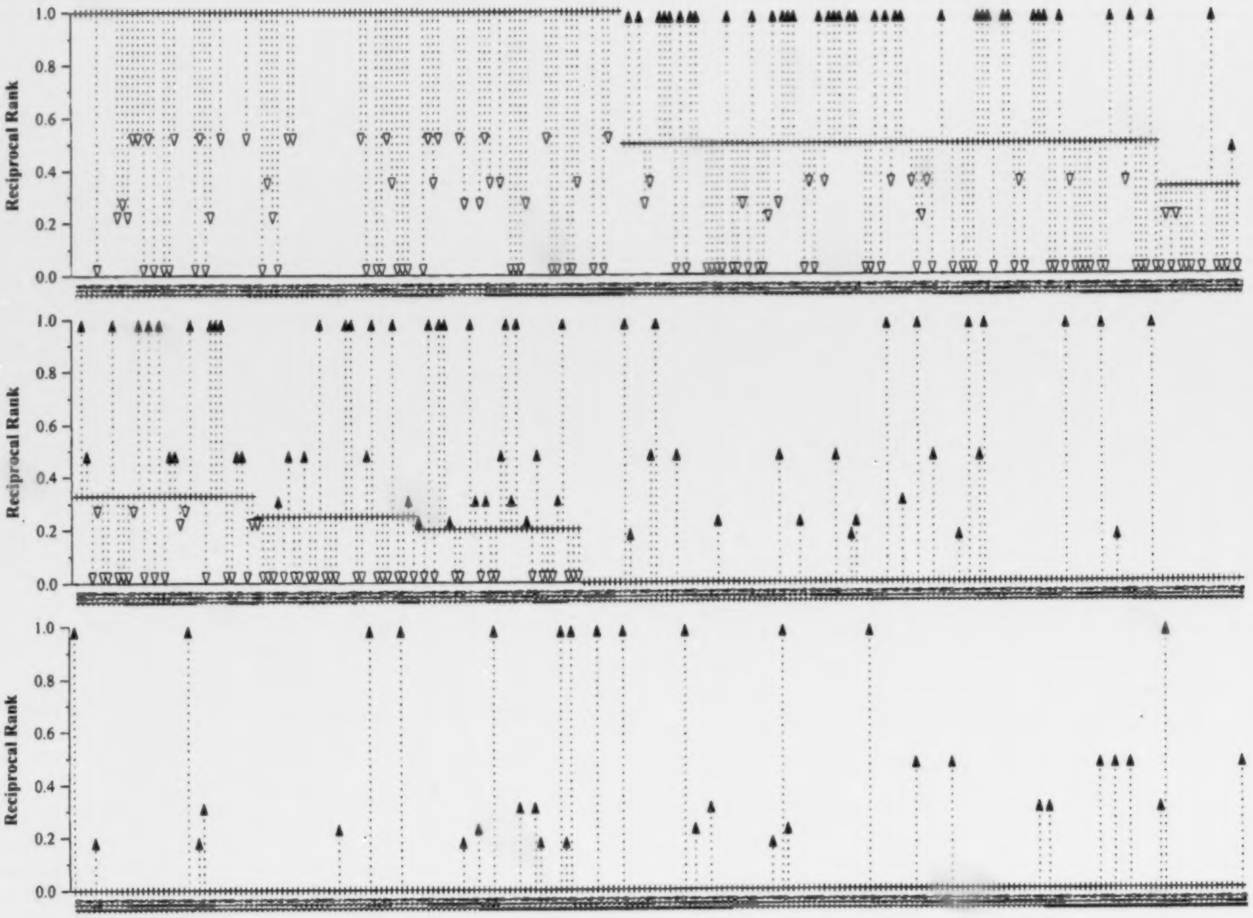
Run:	msq9L250
Category :	250 byte limit
Mean reciprocal rank:	0.264
# questions with no answer found:	417
Average answer length:	68.080 bytes



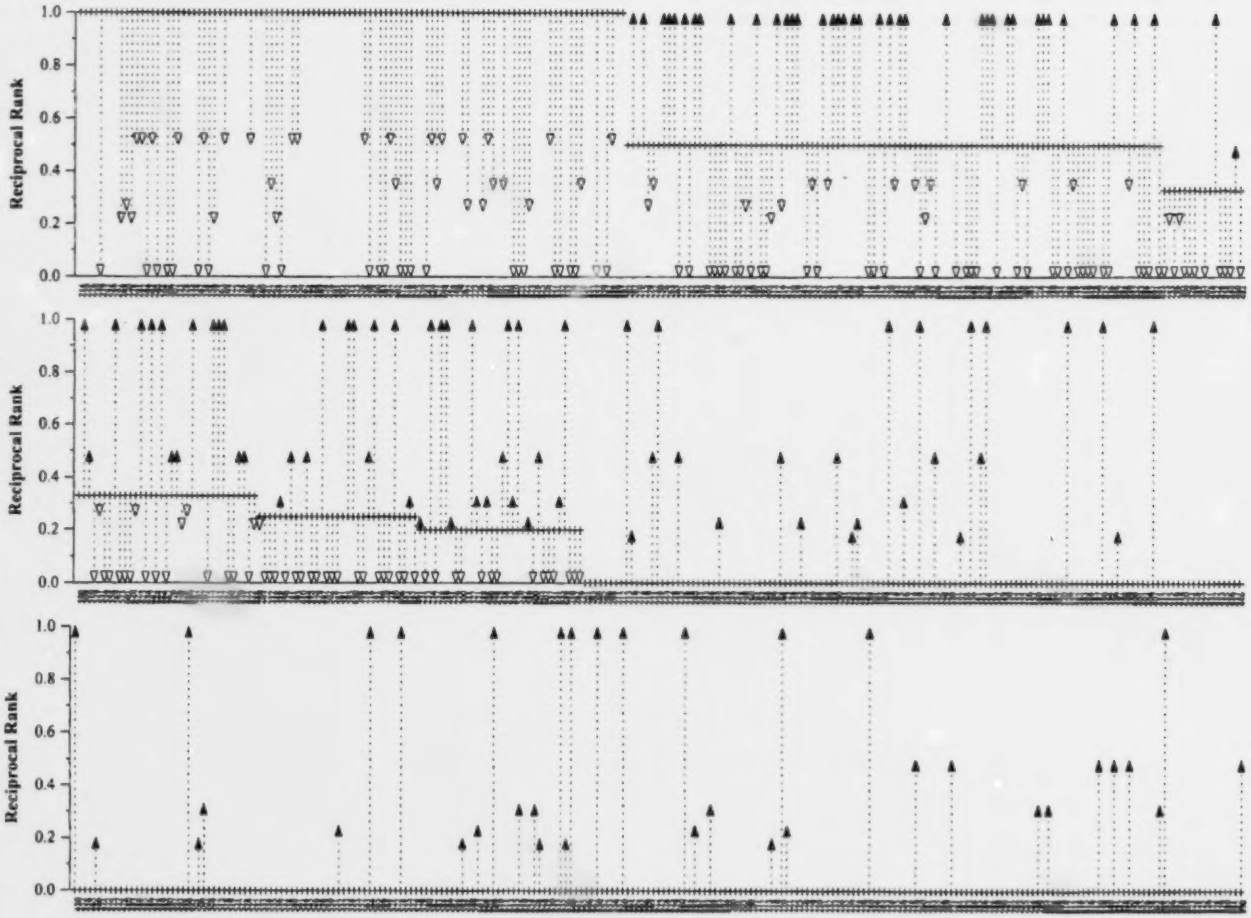
Run:	msq9L250
Category :	250 byte limit
Mean reciprocal rank:	0.264
# questions with no answer found:	417
Average answer length:	68.080 bytes



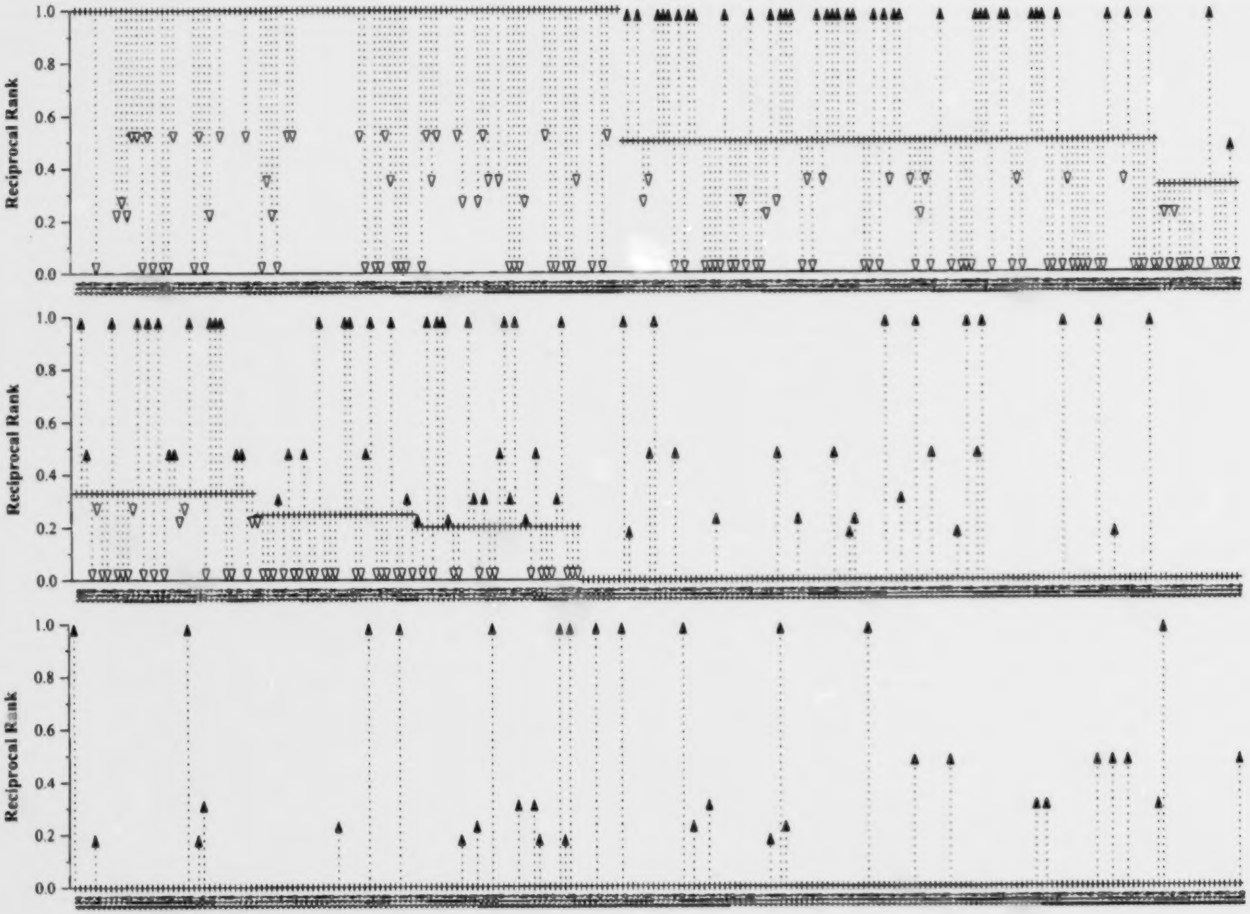
Run:	msq9L250
Category :	250 byte limit
Mean reciprocal rank:	0.264
# questions with no answer found:	417
Average answer length:	68.080 bytes



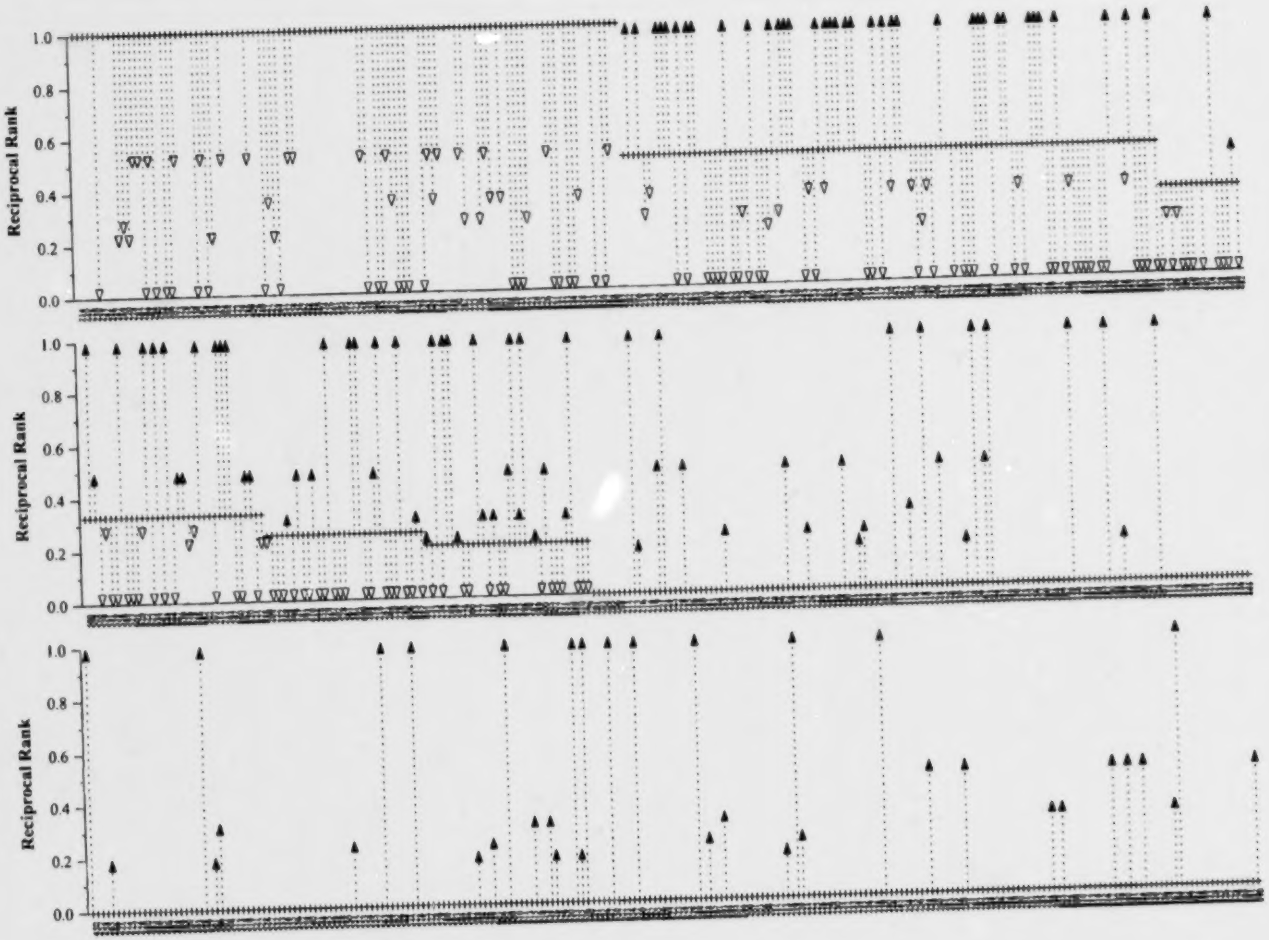
Run:	msq9L250
Category :	250 byte limit
Mean reciprocal rank:	0.264
# questions with no answer found:	417
Average answer length:	68.080 bytes



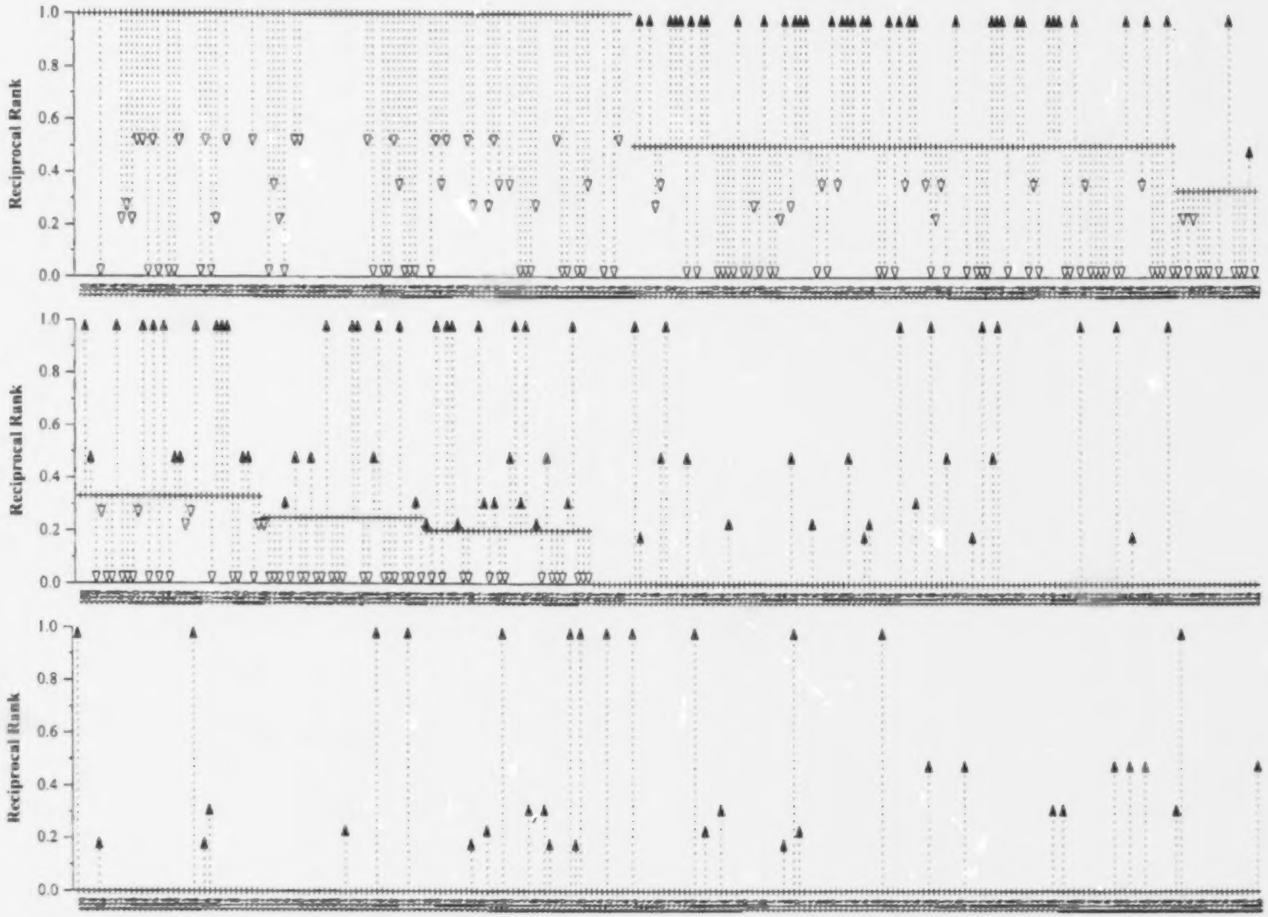
Run:	msq9L250
Category :	250 byte limit
Mean reciprocal rank:	0.264
# questions with no answer found:	417
Average answer length:	68.080 bytes



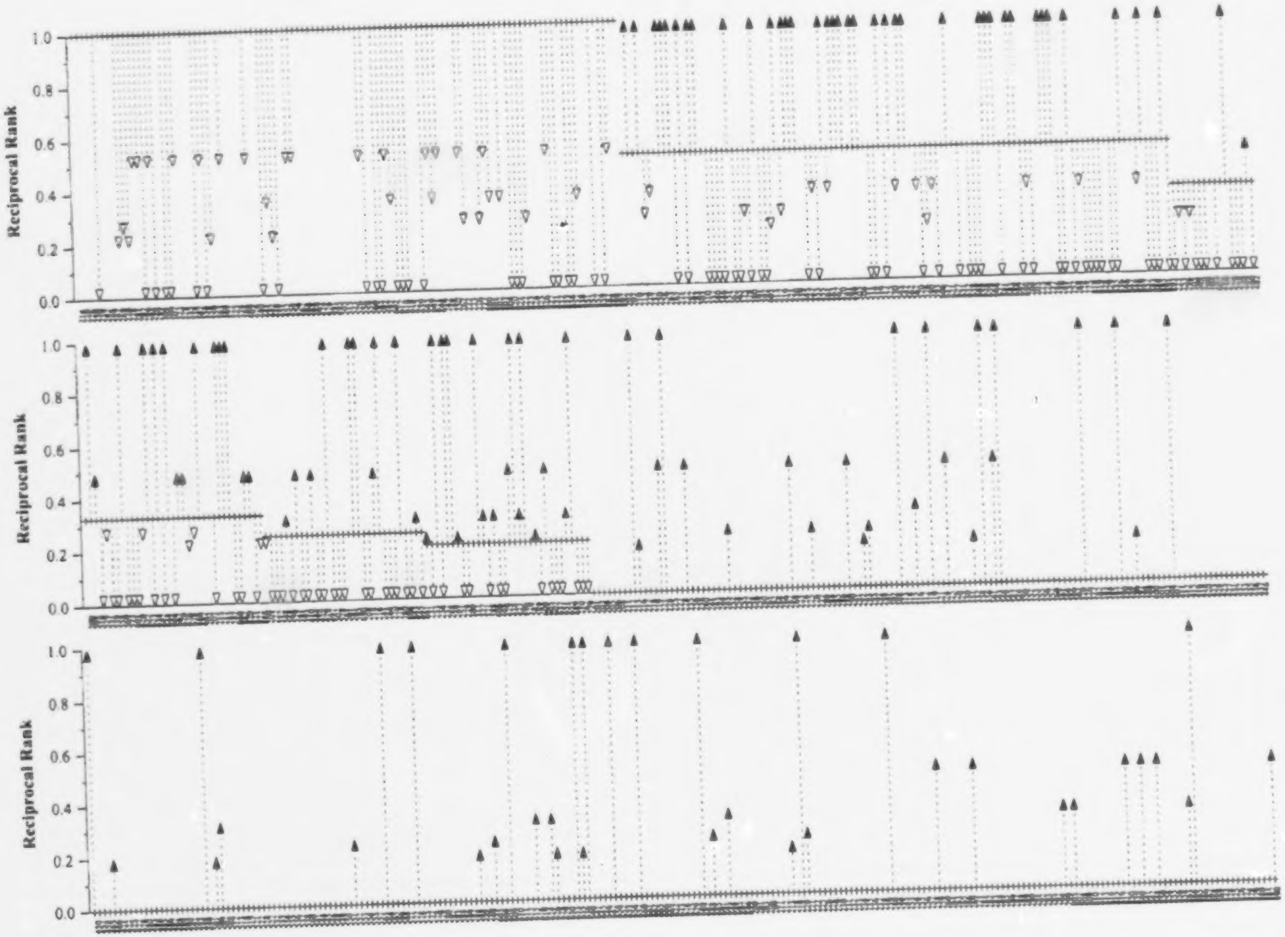
Run:	msq9L250
Category :	250 byte limit
Mean reciprocal rank:	0.264
# questions with no answer found:	417
Average answer length:	68.080 bytes



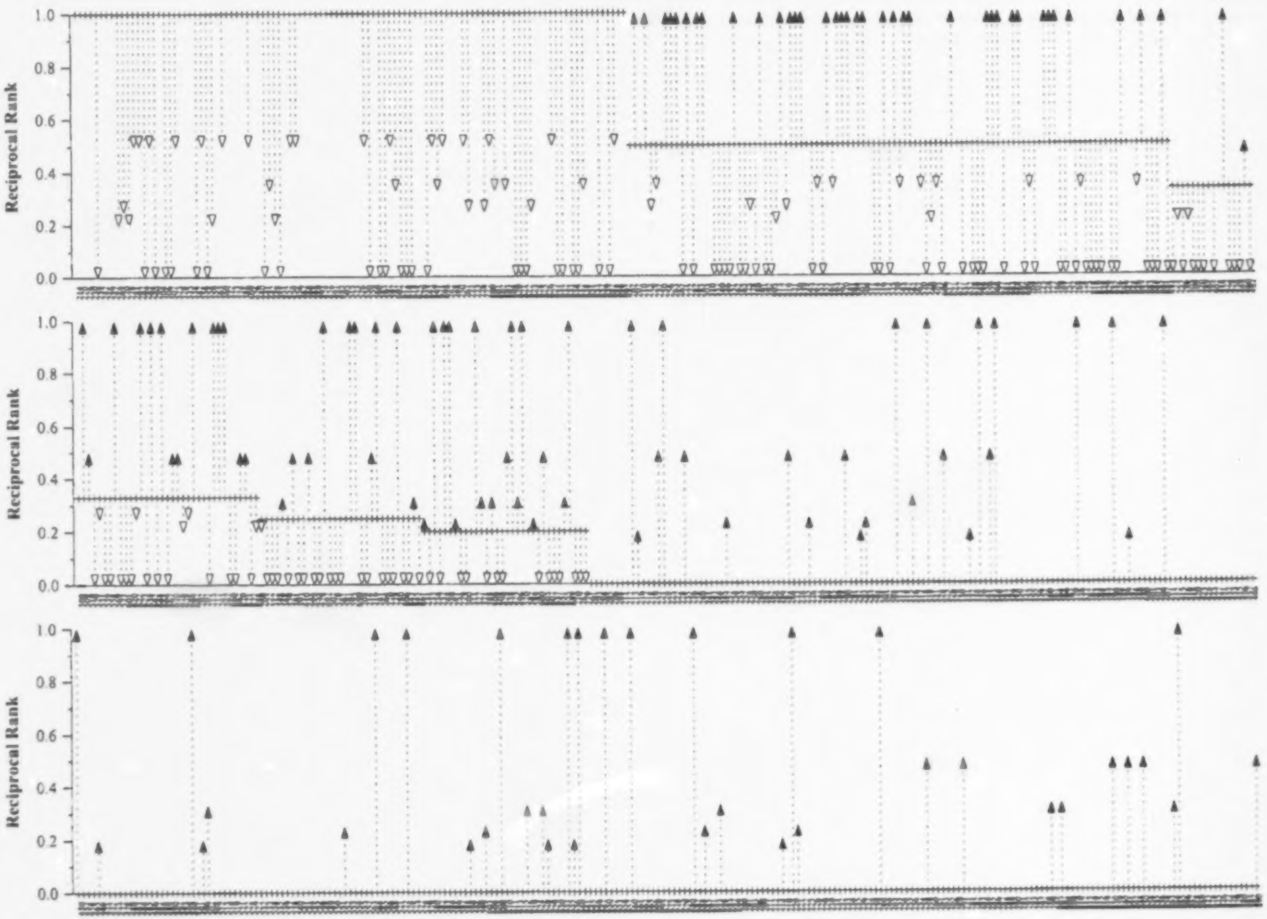
Run:	msq9L250
Category :	250 byte limit
Mean reciprocal rank:	0.264
# questions with no answer found:	417
Average answer length:	68.080 bytes



Run:	msq9L250
Category :	250 byte limit
Mean reciprocal rank:	0.264
# questions with no answer found:	417
Average answer length:	68.080 bytes



Run:	msq9L250
Category :	250 byte limit
Mean reciprocal rank:	0.264
# questions with no answer found:	417
Average answer length:	68.080 bytes



TREC-9 Spoken Document Retrieval Results

Publications home NIST Special Publication XXX-XXX

Help NIST HOME

- cuhtk-b1su
- cuhtk-b1sun
- cuhtk-b1tu
- cuhtk-b1tun
- cuhtk-crsu-cuhtk99s1p1u
- cuhtk-crsu-limsi1u
- cuhtk-crsu-limsi2u
- cuhtk-crsu-nist99b1u
- cuhtk-crsu-shef1u
- cuhtk-crsu-shef2u
- cuhtk-crtu-cuhtk99s1p1u
- cuhtk-crtu-limsi1u
- cuhtk-crtu-limsi2u
- cuhtk-crtu-nist99b1u
- cuhtk-crtu-shef1u
- cuhtk-crtu-shef2u
- cuhtk-r1su
- cuhtk-r1sun
- cuhtk-r1tu

- cuhtk-r1tun
- cuhtk-s1su
- cuhtk-s1sun
- cuhtk-s1tu
- cuhtk-s1tun
- shef-b1tk
- shef-b1tu
- cuhtk-r1tk
- cuhtk-r1sk
- cuhtk-s1sk
- cuhtk-s1tk
- shef-crsu-cuhtk1p1u
- shef-crsu-cuhtk1u
- shef-crsu-limsi2u
- shef-crtu-limsi2u
- shef-crtu-limsi1u
- shef-crtu-cuhtk1u
- shef-crtu-cuhtk1p1u
- shef-r1sk
- shef-r1su
- shef-r1tk
- shef-r1tu
- shef-s1sk-shef1u
- shef-s1su-shef1u
- shef-s1tk-shef1u

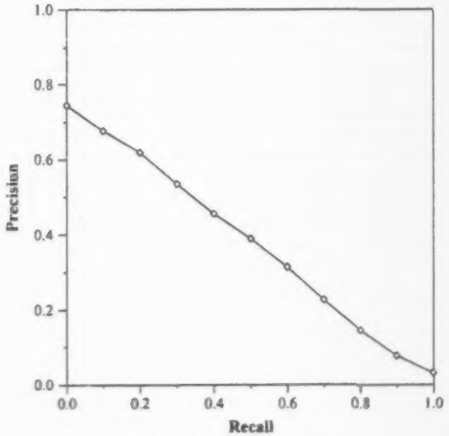
- shef-s1tu-shef1u
- cuhtk-b1tk
- cuhtk-b1sk
- shef-s2tu-shef2u
- shef-s2tk-shef2u
- shef-s2sk-shef2u
- shef-s2su-shef2u
- shef-crsu-limsi1u
- shef-b1su
- shef-b1sk
- limsi-r1sk
- limsi-r1su
- limsi-r1tk
- limsi-r1tu
- limsi-s1sk-limsi2u
- limsi-s1su-limsi2u
- limsi-s1tk-limsi2u
- limsi-s1tu-limsi2u
- limsi-b1su
- limsi-b1tu

Last updated: Tuesday, 28-Aug-01 09:52:01
Date created: Tuesday, 28-Aug-01
trec@nist.gov

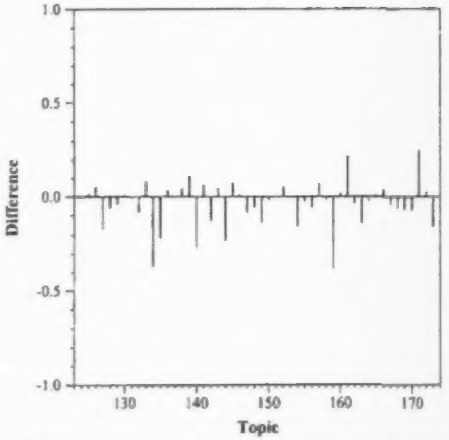
Summary Statistics	
Run Number	cuhtk-b1su
Run Description	unknown, fixed, short, non-lexical info used
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2216
Rel-ret:	1654

Recall Level Precision Averages	
Recall	Precision
0.00	0.7447
0.10	0.6775
0.20	0.6189
0.30	0.5353
0.40	0.4570
0.50	0.3892
0.60	0.3141
0.70	0.2269
0.80	0.1454
0.90	0.0774
1.00	0.0316
Average precision over all relevant docs	
non-interpolated	0.3708

Document Level Averages	
	Precision
At 5 docs	0.5800
At 10 docs	0.5440
At 15 docs	0.4947
At 20 docs	0.4600
At 30 docs	0.3873
At 100 docs	0.2104
At 200 docs	0.1301
At 500 docs	0.0626
At 1000 docs	0.0331
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3991



Recall-Precision Curve

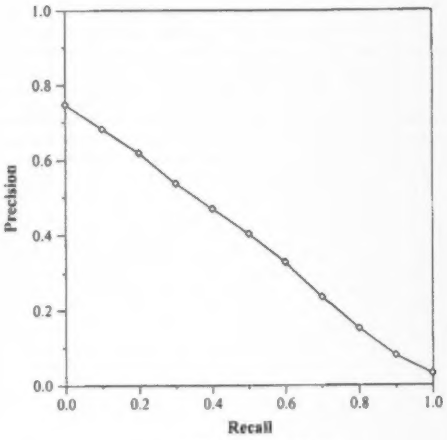


Difference from Median in Average Precision per Topic

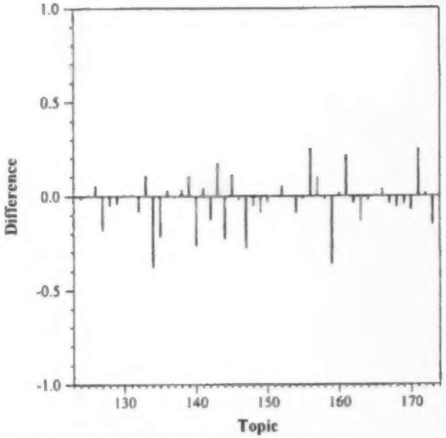
Summary Statistics	
Run Number	cuhtk-b1sun
Run Description	unknown, fixed, short
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2216
Rel-ret:	1682

Recall Level Precision Averages	
Recall	Precision
0.00	0.7488
0.10	0.6821
0.20	0.6190
0.30	0.5392
0.40	0.4704
0.50	0.4032
0.60	0.3290
0.70	0.2358
0.80	0.1511
0.90	0.0802
1.00	0.0316
Average precision over all relevant docs	
non-interpolated	0.3781

Document Level Averages	
	Precision
At 5 docs	0.5840
At 10 docs	0.5420
At 15 docs	0.5027
At 20 docs	0.4670
At 30 docs	0.3933
At 100 docs	0.2126
At 200 docs	0.1314
At 500 docs	0.0637
At 1000 docs	0.0336
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4044



Recall-Precision Curve



Difference from Median in Average Precision per Topic

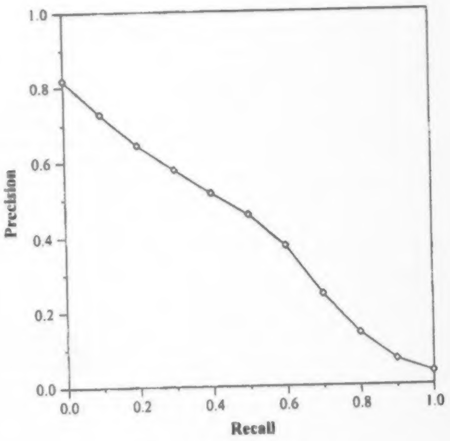
Spoken document retrieval track results — Cambridge University

Summary Statistics	
Run Number	cuhtk-b1tu
Run Description	unknown, fixed, terse, non-lexical info used
Number of Topics	50

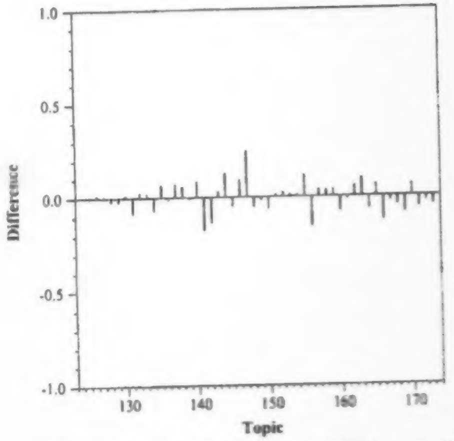
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2216
Rel-ret:	1721

Recall Level Precision Averages	
Recall	Precision
0.00	0.8180
0.10	0.7283
0.20	0.6454
0.30	0.5785
0.40	0.5162
0.50	0.4586
0.60	0.3746
0.70	0.2448
0.80	0.1408
0.90	0.0686
1.00	0.0338
Average precision over all relevant docs	
non-interpolated	0.4075

Document Level Averages	
	Precision
At 5 docs	0.6480
At 10 docs	0.5740
At 15 docs	0.5293
At 20 docs	0.4880
At 30 docs	0.4120
At 100 docs	0.2266
At 200 docs	0.1415
At 500 docs	0.0662
At 1000 docs	0.0344
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4387



Recall-Precision Curve

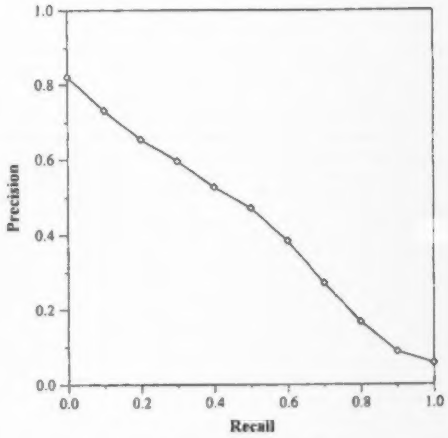


Difference from Median in Average Precision per Topic

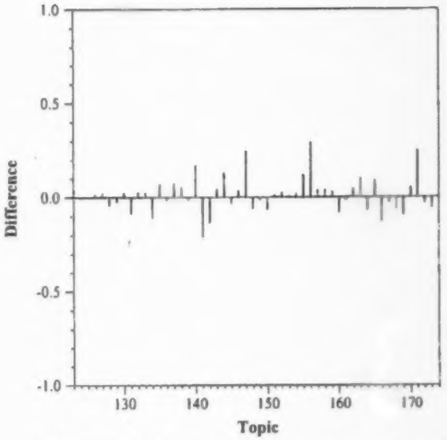
Summary Statistics	
Run Number	cuhtk-b1tun
Run Description	unknown, fixed, terse
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2216
Rel-ret:	1744

Recall Level Precision Averages	
Recall	Precision
0.00	0.8212
0.10	0.7324
0.20	0.6544
0.30	0.5966
0.40	0.5277
0.50	0.4723
0.60	0.3835
0.70	0.2717
0.80	0.1671
0.90	0.0879
1.00	0.0576
Average precision over all relevant docs	
non-interpolated	0.4217

Document Level Averages	
	Precision
At 5 docs	0.6600
At 10 docs	0.5800
At 15 docs	0.5440
At 20 docs	0.4950
At 30 docs	0.4193
At 100 docs	0.2268
At 200 docs	0.1428
At 500 docs	0.0670
At 1000 docs	0.0349
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4477



Recall-Precision Curve



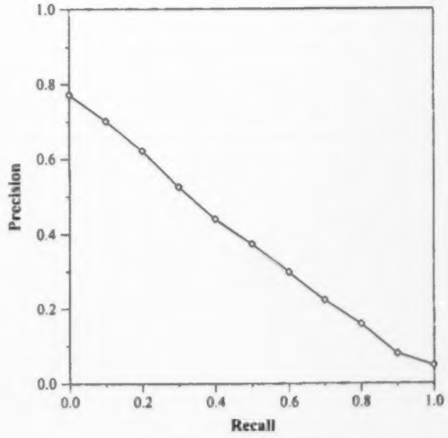
Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	cuhtk-crsu-cuhtk99s1p1u
Run Description	unknown, fixed, short, non-lexical info used
Number of Topics	50

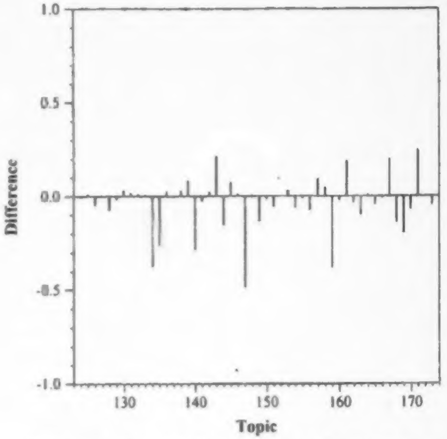
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2216
Rel-ret:	1648

Recall Level Precision Averages	
Recall	Precision
0.00	0.7717
0.10	0.7018
0.20	0.6217
0.30	0.5252
0.40	0.4395
0.50	0.3730
0.60	0.2987
0.70	0.2221
0.80	0.1583
0.90	0.0792
1.00	0.0477
Average precision over all relevant docs	
non-interpolated	0.3726

Document Level Averages	
	Precision
At 5 docs	0.5960
At 10 docs	0.5320
At 15 docs	0.4893
At 20 docs	0.4430
At 30 docs	0.3707
At 100 docs	0.2028
At 200 docs	0.1289
At 500 docs	0.0625
At 1000 docs	0.0330
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3949



Recall-Precision Curve



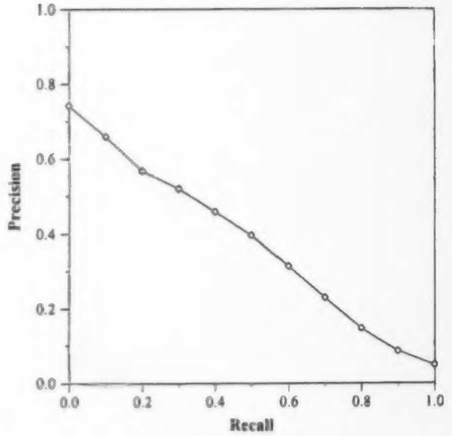
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — Cambridge University

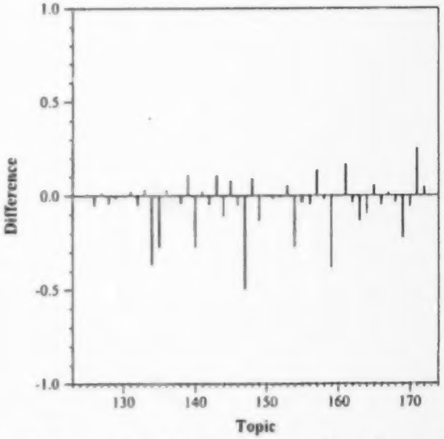
Summary Statistics	
Run Number	cuhtk-crsu-limsi1u
Run Description	unknown, fixed, short, non-lexical info used
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2216
Rel-ret:	1666

Recall Level Precision Averages	
Recall	Precision
0.00	0.7428
0.10	0.6595
0.20	0.5677
0.30	0.5195
0.40	0.4589
0.50	0.3968
0.60	0.3131
0.70	0.2302
0.80	0.1463
0.90	0.0852
1.00	0.0483
Average precision over all relevant docs	
non-interpolated	0.3656

Document Level Averages	
	Precision
At 5 docs	0.5840
At 10 docs	0.5280
At 15 docs	0.4800
At 20 docs	0.4350
At 30 docs	0.3680
At 100 docs	0.2066
At 200 docs	0.1318
At 500 docs	0.0630
At 1000 docs	0.0333
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3857



Recall-Precision Curve



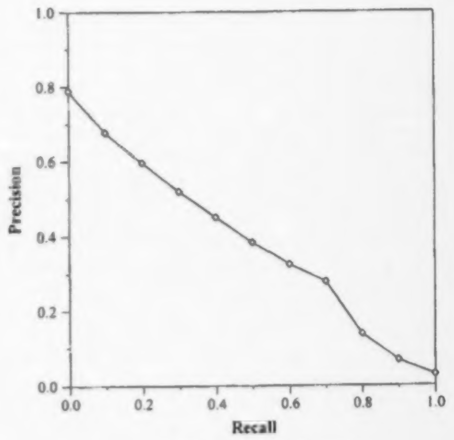
Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	cuhtk-crsu-limsi2u
Run Description	unknown, fixed, short, non-lexical info used
Number of Topics	50

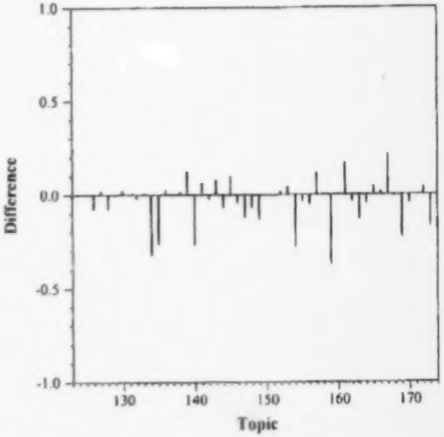
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2216
Rel-ret:	1663

Recall Level Precision Averages	
Recall	Precision
0.00	0.7889
0.10	0.6769
0.20	0.5962
0.30	0.5210
0.40	0.4516
0.50	0.3841
0.60	0.3251
0.70	0.2792
0.80	0.1368
0.90	0.0679
1.00	0.0298
Average precision over all relevant docs	
non-interpolated	0.3724

Document Level Averages	
	Precision
At 5 docs	0.5640
At 10 docs	0.5160
At 15 docs	0.4760
At 20 docs	0.4400
At 30 docs	0.3800
At 100 docs	0.2064
At 200 docs	0.1319
At 500 docs	0.0632
At 1000 docs	0.0333
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3928



Recall-Precision Curve



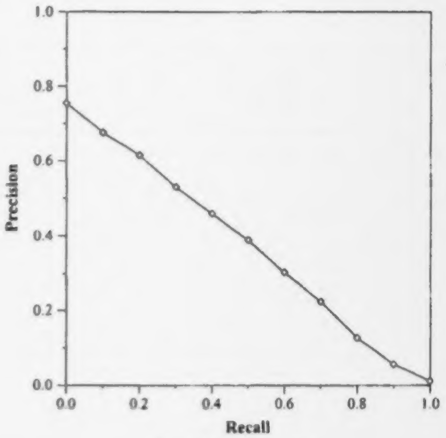
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — Cambridge University

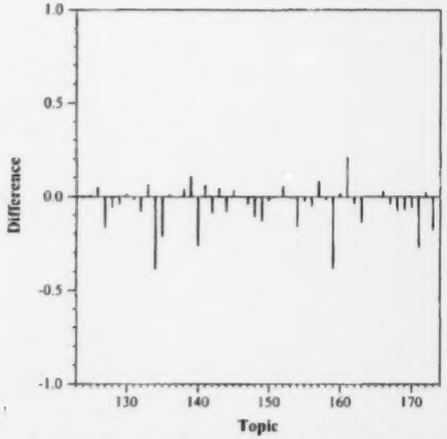
Summary Statistics	
Run Number	cuhtk-crsu-nist99b1u
Run Description	unknown, fixed, short, non-lexical info used
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2216
Rel-ret:	1651

Recall Level Precision Averages	
Recall	Precision
0.00	0.7553
0.10	0.6755
0.20	0.6148
0.30	0.5304
0.40	0.4591
0.50	0.3885
0.60	0.3037
0.70	0.2242
0.80	0.1271
0.90	0.0567
1.00	0.0118
Average precision over all relevant docs	
non-interpolated	0.3608

Document Level Averages	
	Precision
At 5 docs	0.5920
At 10 docs	0.5400
At 15 docs	0.4893
At 20 docs	0.4530
At 30 docs	0.3820
At 100 docs	0.2084
At 200 docs	0.1302
At 500 docs	0.0626
At 1000 docs	0.0330
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3986



Recall-Precision Curve



Difference from Median in Average Precision per Topic

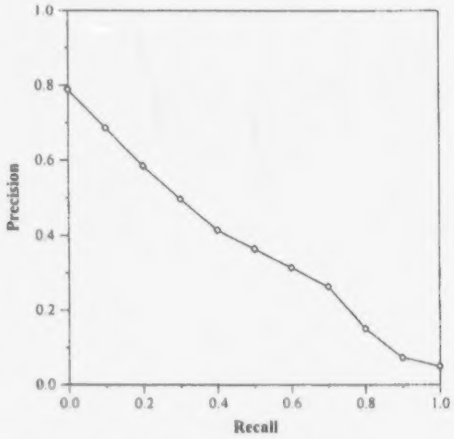
Spoken document retrieval track results — Cambridge University

Summary Statistics	
Run Number	cuhtk-crsu-shef1u
Run Description	unknown, fixed, short, non-lexical info used
Number of Topics	50

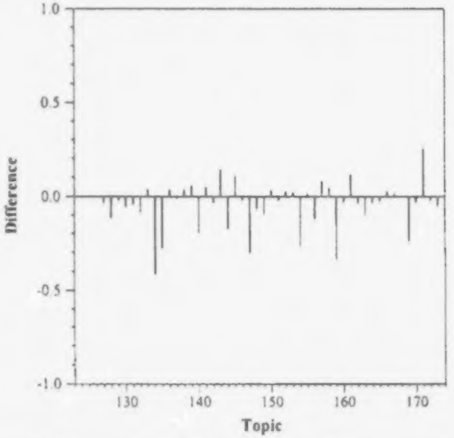
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2216
Rel-ret:	1627

Recall Level Precision Averages	
Recall	Precision
0.00	0.7887
0.10	0.6868
0.20	0.5856
0.30	0.4968
0.40	0.4142
0.50	0.3638
0.60	0.3143
0.70	0.2637
0.80	0.1504
0.90	0.0735
1.00	0.0506
Average precision over all relevant docs	
non-interpolated	0.3644

Document Level Averages	
	Precision
At 5 docs	0.5840
At 10 docs	0.5380
At 15 docs	0.4720
At 20 docs	0.4410
At 30 docs	0.3700
At 100 docs	0.2050
At 200 docs	0.1290
At 500 docs	0.0612
At 1000 docs	0.0325
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3896



Recall-Precision Curve



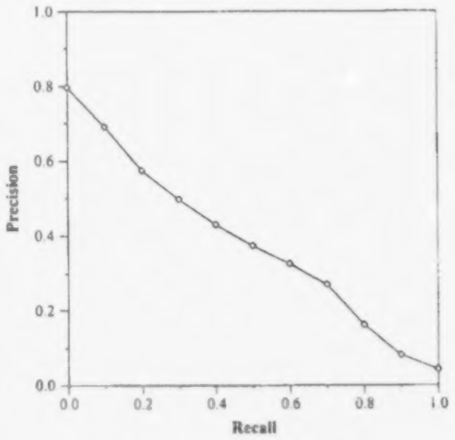
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — Cambridge University

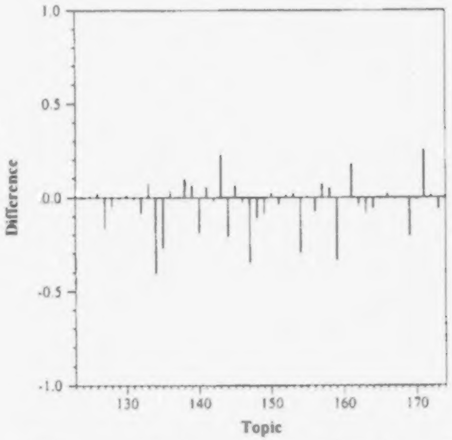
Summary Statistics	
Run Number	cuhtk-crsu-shef2u
Run Description	unknown, fixed, short, non-lexical info used
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2216
Rel-ret:	1672

Recall Level Precision Averages	
Recall	Precision
0.00	0.7975
0.10	0.6900
0.20	0.5744
0.30	0.4977
0.40	0.4285
0.50	0.3731
0.60	0.3258
0.70	0.2698
0.80	0.1614
0.90	0.0818
1.00	0.0425
Average precision over all relevant docs	
non-interpolated	0.3703

Document Level Averages	
	Precision
At 5 docs	0.5880
At 10 docs	0.5420
At 15 docs	0.4813
At 20 docs	0.4410
At 30 docs	0.3793
At 100 docs	0.2032
At 200 docs	0.1308
At 500 docs	0.0630
At 1000 docs	0.0334
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3948



Recall-Precision Curve



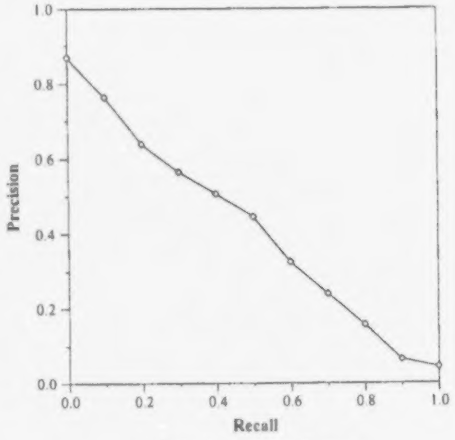
Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	cuhtk-crtu-cuhtk99s1p1u
Run Description	unknown, fixed, terse, non-lexical info used
Number of Topics	50

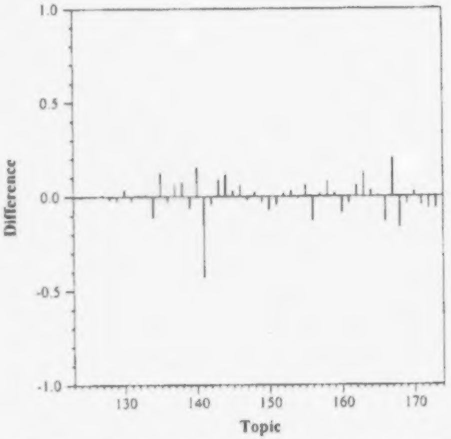
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2216
Rel-ret:	1701

Recall Level Precision Averages	
Recall	Precision
0.00	0.8696
0.10	0.7637
0.20	0.6384
0.30	0.5650
0.40	0.5064
0.50	0.4454
0.60	0.3257
0.70	0.2392
0.80	0.1568
0.90	0.0652
1.00	0.0439
Average precision over all relevant docs	
non-interpolated	0.4044

Document Level Averages	
	Precision
At 5 docs	0.6320
At 10 docs	0.5800
At 15 docs	0.5320
At 20 docs	0.4810
At 30 docs	0.4073
At 100 docs	0.2212
At 200 docs	0.1397
At 500 docs	0.0651
At 1000 docs	0.0340
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4292



Recall-Precision Curve



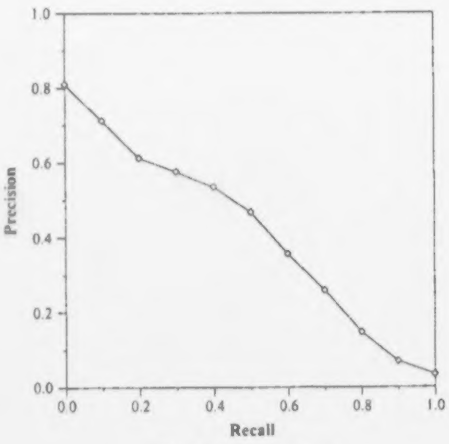
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — Cambridge University

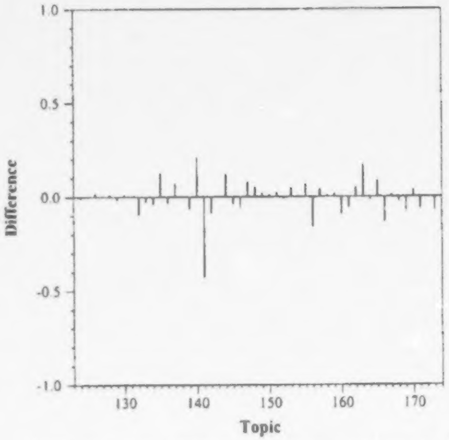
Summary Statistics	
Run Number	cuhtk-crtu-limsi1u
Run Description	unknown, fixed, terse, non-lexical info used
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2216
Rel-ret:	1726

Recall Level Precision Averages	
Recall	Precision
0.00	0.8098
0.10	0.7123
0.20	0.6125
0.30	0.5760
0.40	0.5359
0.50	0.4680
0.60	0.3579
0.70	0.2594
0.80	0.1488
0.90	0.0698
1.00	0.0344
Average precision over all relevant docs	
non-interpolated	0.4019

Document Level Averages	
	Precision
At 5 docs	0.6160
At 10 docs	0.5420
At 15 docs	0.5213
At 20 docs	0.4770
At 30 docs	0.4080
At 100 docs	0.2210
At 200 docs	0.1414
At 500 docs	0.0662
At 1000 docs	0.0345
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4368



Recall-Precision Curve



Difference from Median in Average Precision per Topic

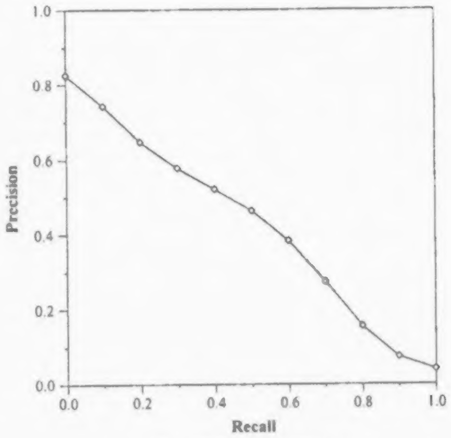
Spoken document retrieval track results — Cambridge University

Summary Statistics	
Run Number	cuhtk-crtu-limsi2u
Run Description	unknown, fixed, terse, non-lexical info used
Number of Topics	50

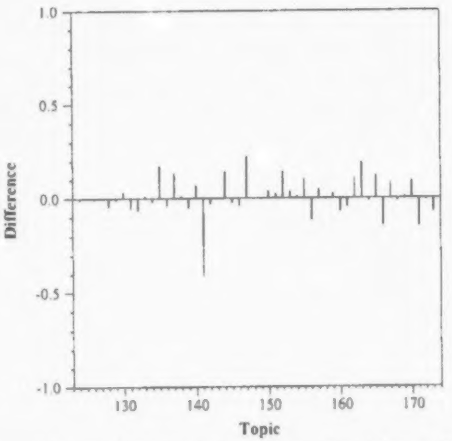
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2216
Rel-ret:	1728

Recall Level Precision Averages	
Recall	Precision
0.00	0.8261
0.10	0.7427
0.20	0.6471
0.30	0.5783
0.40	0.5207
0.50	0.4628
0.60	0.3849
0.70	0.2743
0.80	0.1563
0.90	0.0739
1.00	0.0399
Average precision over all relevant docs	
non-interpolated	0.4162

Document Level Averages	
	Precision
At 5 docs	0.6520
At 10 docs	0.5840
At 15 docs	0.5387
At 20 docs	0.4950
At 30 docs	0.4173
At 100 docs	0.2254
At 200 docs	0.1445
At 500 docs	0.0664
At 1000 docs	0.0346
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4412



Recall-Precision Curve



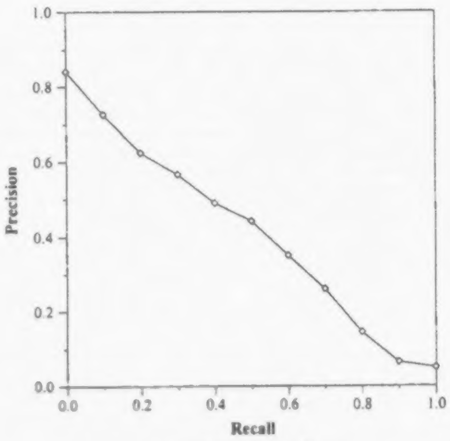
Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	cuhtk-crtu-shef1u
Run Description	unknown, fixed, terse, non-lexical info used
Number of Topics	50

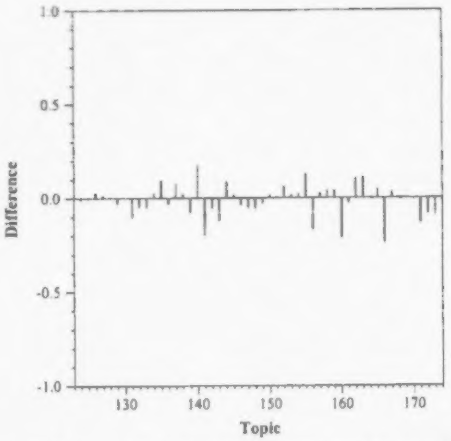
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2216
Rel-ret:	1687

Recall Level Precision Averages	
Recall	Precision
0.00	0.8419
0.10	0.7270
0.20	0.6248
0.30	0.5663
0.40	0.4900
0.50	0.4423
0.60	0.3499
0.70	0.2596
0.80	0.1448
0.90	0.0646
1.00	0.0499
Average precision over all relevant docs	
non-interpolated	0.3958

Document Level Averages	
	Precision
At 5 docs	0.6320
At 10 docs	0.5640
At 15 docs	0.5173
At 20 docs	0.4750
At 30 docs	0.4113
At 100 docs	0.2226
At 200 docs	0.1411
At 500 docs	0.0643
At 1000 docs	0.0337
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4242



Recall-Precision Curve



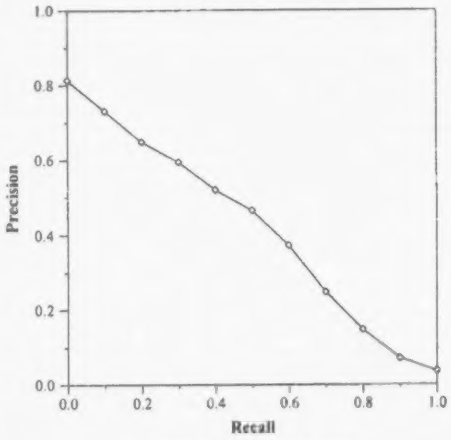
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — Cambridge University

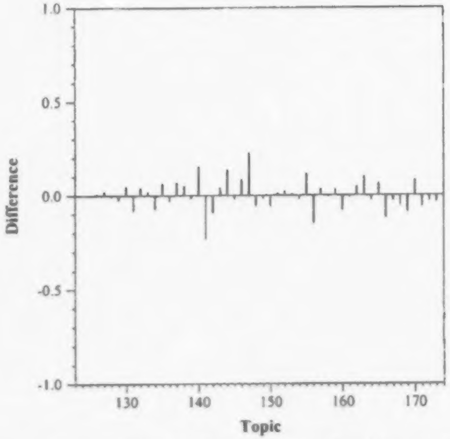
Summary Statistics	
Run Number	cuhtk-crtu-nist99b1u
Run Description	unknown, fixed, terse, non-lexical info used
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2216
Rel-ret:	1723

Recall Level Precision Averages	
Recall	Precision
0.00	0.8136
0.10	0.7310
0.20	0.6493
0.30	0.5947
0.40	0.5197
0.50	0.4658
0.60	0.3727
0.70	0.2473
0.80	0.1480
0.90	0.0706
1.00	0.0363
Average precision over all relevant docs	
non-interpolated	0.4099

Document Level Averages	
	Precision
At 5 docs	0.6600
At 10 docs	0.5720
At 15 docs	0.5293
At 20 docs	0.4870
At 30 docs	0.4140
At 100 docs	0.2260
At 200 docs	0.1417
At 500 docs	0.0662
At 1000 docs	0.0345
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4439



Recall-Precision Curve



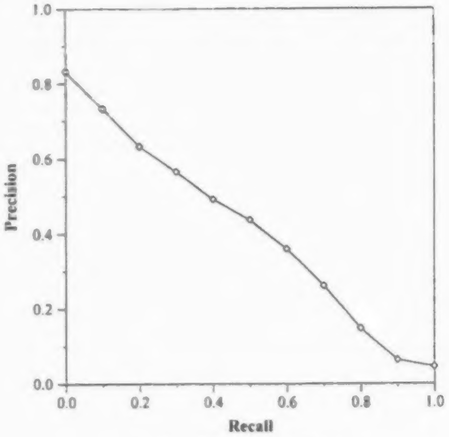
Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	cuhtk-crtu-shef2u
Run Description	unknown, fixed, terse, non-lexical info used
Number of Topics	50

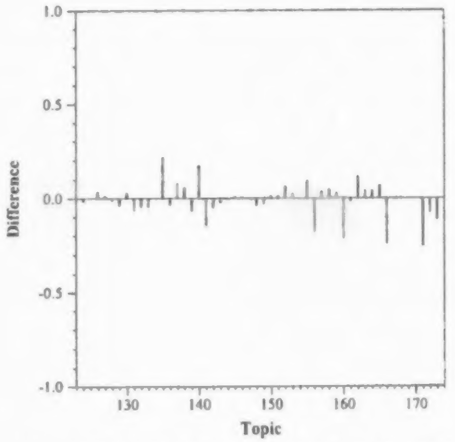
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2216
Rel-ret:	1711

Recall Level Precision Averages	
Recall	Precision
0.00	0.8335
0.10	0.7325
0.20	0.6341
0.30	0.5667
0.40	0.4922
0.50	0.4379
0.60	0.3610
0.70	0.2617
0.80	0.1489
0.90	0.0634
1.00	0.0456
Average precision over all relevant docs	
non-interpolated	0.3983

Document Level Averages	
	Precision
At 5 docs	0.6040
At 10 docs	0.5640
At 15 docs	0.5253
At 20 docs	0.4730
At 30 docs	0.4100
At 100 docs	0.2254
At 200 docs	0.1429
At 500 docs	0.0653
At 1000 docs	0.0342
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4265



Recall-Precision Curve



Difference from Median in Average Precision per Topic

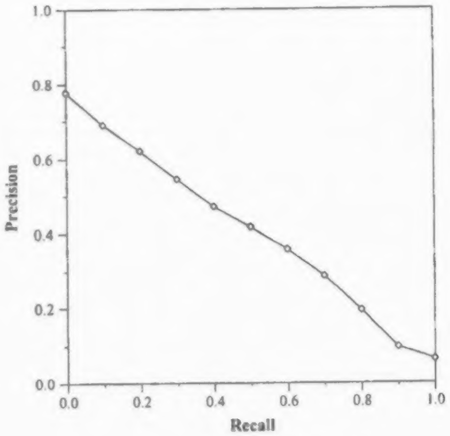
Spoken document retrieval track results — Cambridge University

Summary Statistics	
Run Number	cuhtk-r1su
Run Description	unknown, fixed, short, non-lexical info used
Number of Topics	50

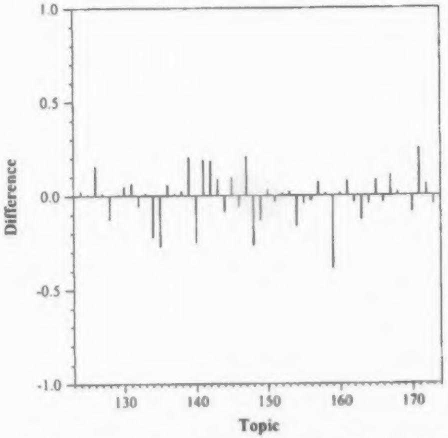
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2216
Rel-ret:	1703

Recall Level Precision Averages	
Recall	Precision
0.00	0.7779
0.10	0.6891
0.20	0.6217
0.30	0.5453
0.40	0.4714
0.50	0.4179
0.60	0.3595
0.70	0.2868
0.80	0.1947
0.90	0.0970
1.00	0.0637
Average precision over all relevant docs	
non-interpolated	0.4003

Document Level Averages	
	Precision
At 5 docs	0.6080
At 10 docs	0.5520
At 15 docs	0.4933
At 20 docs	0.4510
At 30 docs	0.3847
At 100 docs	0.2122
At 200 docs	0.1352
At 500 docs	0.0643
At 1000 docs	0.0341
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4209



Recall-Precision Curve

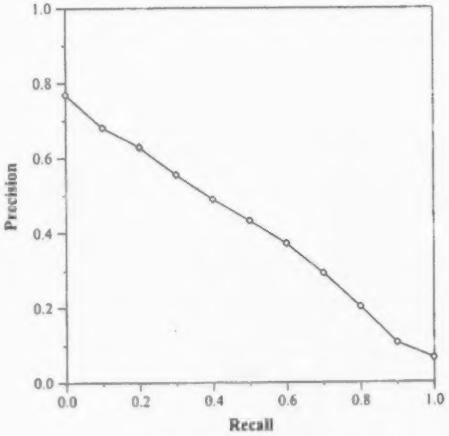


Difference from Median in Average Precision per Topic

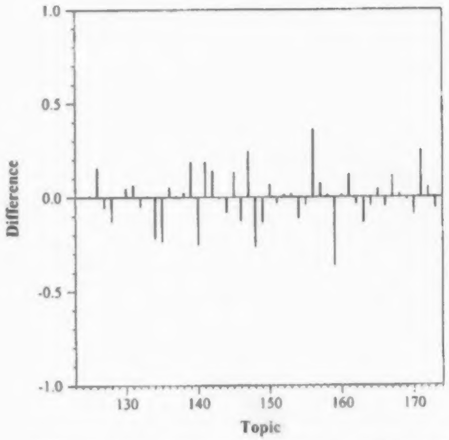
Summary Statistics	
Run Number	cuhtk-r1sun
Run Description	unknown, fixed, short
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2216
Rel-ret:	1726

Recall Level Precision Averages	
Recall	Precision
0.00	0.7702
0.10	0.6804
0.20	0.6283
0.30	0.5559
0.40	0.4898
0.50	0.4329
0.60	0.3724
0.70	0.2929
0.80	0.2023
0.90	0.1079
1.00	0.0667
Average precision over all relevant docs	
non-interpolated	0.4054

Document Level Averages	
	Precision
At 5 docs	0.6040
At 10 docs	0.5500
At 15 docs	0.5000
At 20 docs	0.4560
At 30 docs	0.3913
At 100 docs	0.2096
At 200 docs	0.1352
At 500 docs	0.0648
At 1000 docs	0.0345
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4250



Recall-Precision Curve



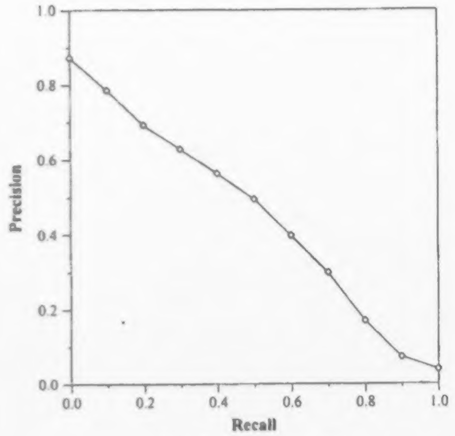
Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	cuhtk-r1tu
Run Description	unknown, fixed, terse, non-lexical info used
Number of Topics	50

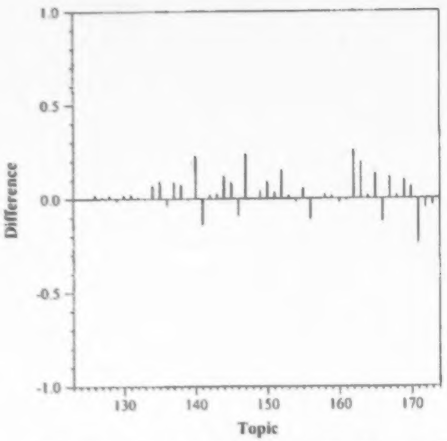
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2216
Rel-ret:	1758

Recall Level Precision Averages	
Recall	Precision
0.00	0.8730
0.10	0.7860
0.20	0.6925
0.30	0.6279
0.40	0.5635
0.50	0.4951
0.60	0.3969
0.70	0.2982
0.80	0.1683
0.90	0.0718
1.00	0.0385
Average precision over all relevant docs	
non-interpolated	0.4402

Document Level Averages	
	Precision
At 5 docs	0.6520
At 10 docs	0.6060
At 15 docs	0.5600
At 20 docs	0.5130
At 30 docs	0.4393
At 100 docs	0.2312
At 200 docs	0.1464
At 500 docs	0.0671
At 1000 docs	0.0352
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4738



Recall-Precision Curve



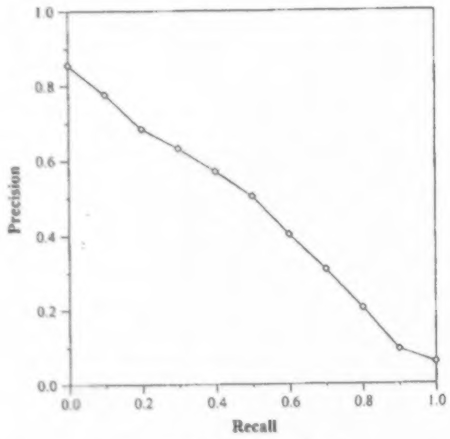
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — Cambridge University

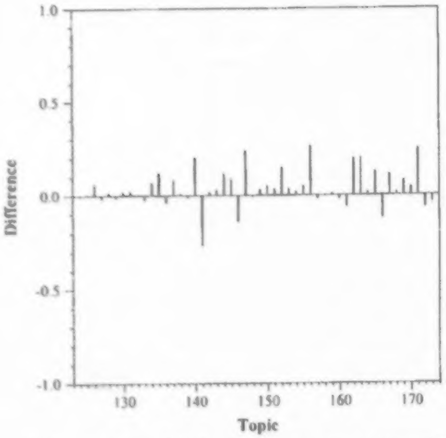
Summary Statistics	
Run Number	cuhtk-r1tun
Run Description	unknown, fixed, terse
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2216
Rel-ret:	1770

Recall Level Precision Averages	
Recall	Precision
0.00	0.8559
0.10	0.7765
0.20	0.6840
0.30	0.6318
0.40	0.5697
0.50	0.5033
0.60	0.4017
0.70	0.3079
0.80	0.2035
0.90	0.0934
1.00	0.0585
Average precision over all relevant docs	
non-interpolated	0.4475

Document Level Averages	
	Precision
At 5 docs	0.6400
At 10 docs	0.6000
At 15 docs	0.5680
At 20 docs	0.5130
At 30 docs	0.4373
At 100 docs	0.2284
At 200 docs	0.1472
At 500 docs	0.0677
At 1000 docs	0.0354
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4703



Recall-Precision Curve

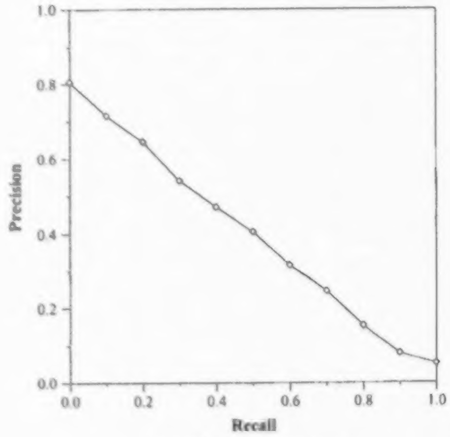


Difference from Median in Average Precision per Topic

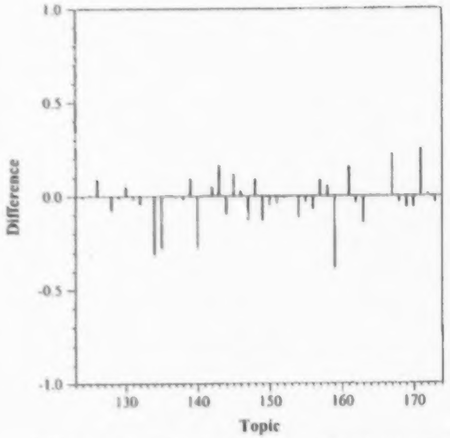
Summary Statistics	
Run Number	cuhtk-s1su
Run Description	unknown, fixed, short, non-lexical info used
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2216
Rel-ret:	1670

Recall Level Precision Averages	
Recall	Precision
0.00	0.8062
0.10	0.7156
0.20	0.6468
0.30	0.5413
0.40	0.4708
0.50	0.4058
0.60	0.3153
0.70	0.2462
0.80	0.1540
0.90	0.0800
1.00	0.0518
Average precision over all relevant docs	
non-interpolated	0.3883

Document Level Averages	
	Precision
At 5 docs	0.6080
At 10 docs	0.5420
At 15 docs	0.4773
At 20 docs	0.4410
At 30 docs	0.3753
At 100 docs	0.2050
At 200 docs	0.1315
At 500 docs	0.0630
At 1000 docs	0.0334
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4036



Recall-Precision Curve

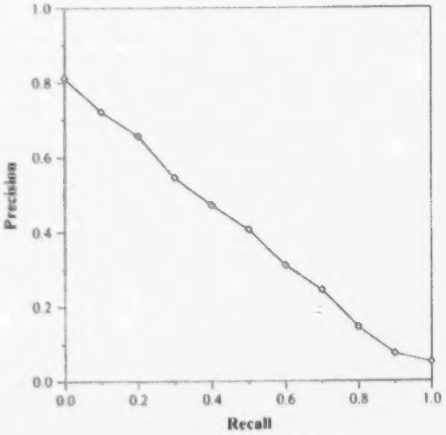


Difference from Median in Average Precision per Topic

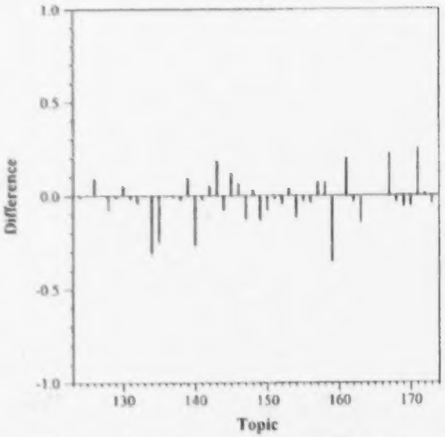
Summary Statistics	
Run Number	cuhtk-s1sun
Run Description	unknown, fixed, short
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2216
Rel-ret:	1668

Recall Level Precision Averages	
Recall	Precision
0.00	0.8127
0.10	0.7209
0.20	0.6571
0.30	0.5456
0.40	0.4710
0.50	0.4071
0.60	0.3116
0.70	0.2443
0.80	0.1460
0.90	0.0753
1.00	0.0517
Average precision over all relevant docs	
non-interpolated	0.3900

Document Level Averages	
	Precision
At 5 docs	0.6120
At 10 docs	0.5460
At 15 docs	0.4827
At 20 docs	0.4410
At 30 docs	0.3787
At 100 docs	0.2074
At 200 docs	0.1323
At 500 docs	0.0630
At 1000 docs	0.0334
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4035



Recall-Precision Curve



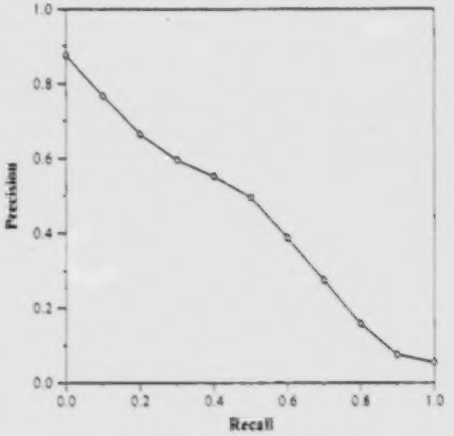
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — Cambridge University

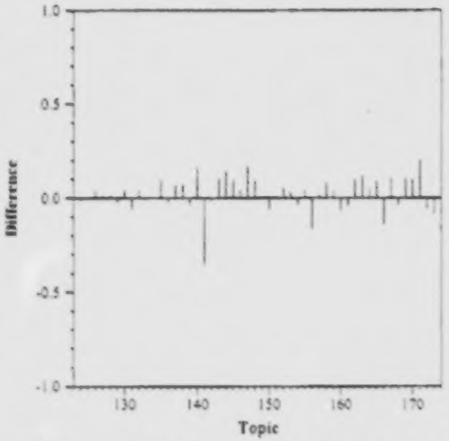
Summary Statistics	
Run Number	cuhtk-s1tu
Run Description	unknown, fixed, terse, non-lexical info used
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2216
Rel-ret:	1739

Recall Level Precision Averages	
Recall	Precision
0.00	0.8757
0.10	0.7672
0.20	0.6631
0.30	0.5952
0.40	0.5528
0.50	0.4951
0.60	0.3867
0.70	0.2747
0.80	0.1570
0.90	0.0743
1.00	0.0547
Average precision over all relevant docs	
non-interpolated	0.4299

Document Level Averages	
	Precision
At 5 docs	0.6560
At 10 docs	0.5940
At 15 docs	0.5480
At 20 docs	0.4930
At 30 docs	0.4173
At 100 docs	0.2290
At 200 docs	0.1445
At 500 docs	0.0668
At 1000 docs	0.0348
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4502



Recall-Precision Curve



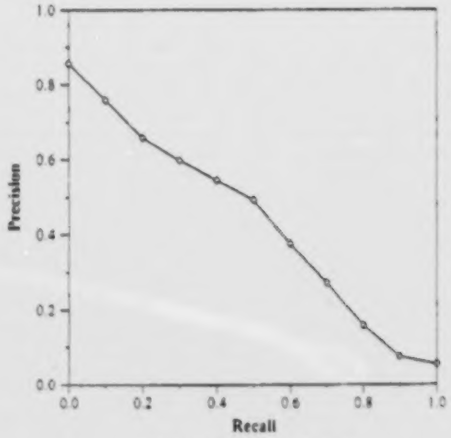
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — Cambridge University

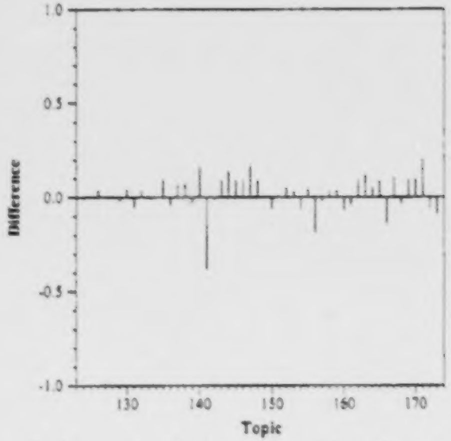
Summary Statistics	
Run Number	cuhtk-s1tun
Run Description	unknown, fixed, terse
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2216
Rel-ret:	1745

Recall Level Precision Averages	
Recall	Precision
0.00	0.8565
0.10	0.7601
0.20	0.6587
0.30	0.5976
0.40	0.5447
0.50	0.4928
0.60	0.3747
0.70	0.2715
0.80	0.1588
0.90	0.0739
1.00	0.0545
Average precision over all relevant docs	
non-interpolated	0.4265

Document Level Averages	
	Precision
At 5 docs	0.6400
At 10 docs	0.5940
At 15 docs	0.5493
At 20 docs	0.4920
At 30 docs	0.4173
At 100 docs	0.2290
At 200 docs	0.1443
At 500 docs	0.0670
At 1000 docs	0.0349
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4511



Recall-Precision Curve



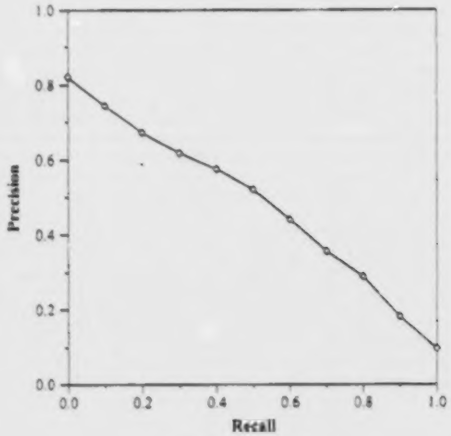
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — U.Sheffield/U.Cambridge/SoftSound/ICSI

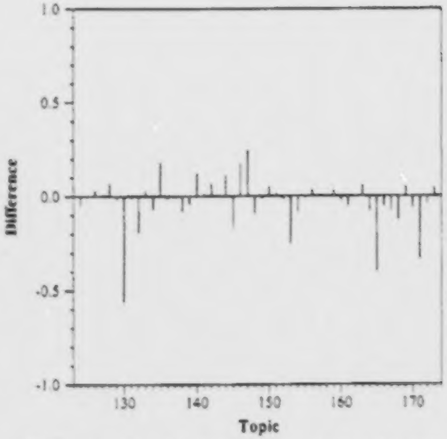
Summary Statistics	
Run Number	shef-b1tk
Run Description	known, rolling, terse
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2123
Rel-ret:	1827

Recall Level Precision Averages	
Recall	Precision
0.00	0.8220
0.10	0.7444
0.20	0.6717
0.30	0.6173
0.40	0.5754
0.50	0.5218
0.60	0.4400
0.70	0.3541
0.80	0.2882
0.90	0.1812
1.00	0.0939
Average precision over all relevant docs	
non-interpolated	0.4689

Document Level Averages	
	Precision
At 5 docs	0.6240
At 10 docs	0.5960
At 15 docs	0.5400
At 20 docs	0.5040
At 30 docs	0.4253
At 100 docs	0.2270
At 200 docs	0.1436
At 500 docs	0.0682
At 1000 docs	0.0365
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4510



Recall-Precision Curve



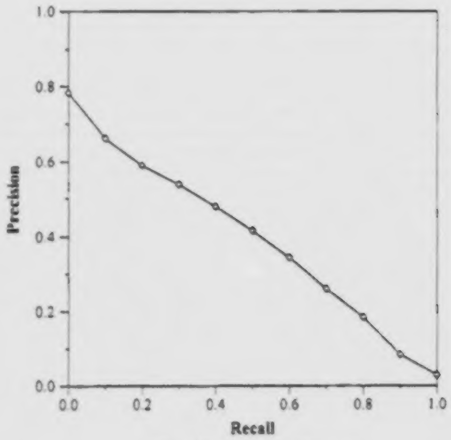
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — U.Sheffield/U.Cambridge/SoftSound/ICSI

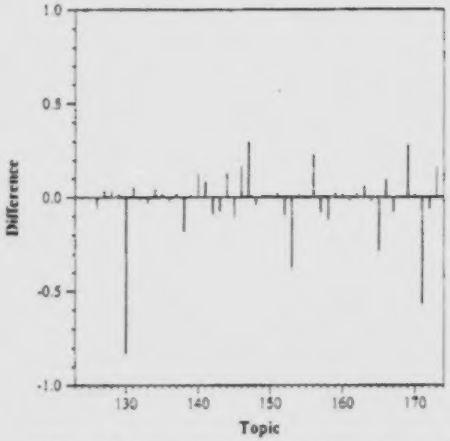
Summary Statistics	
Run Number	shef-b1tu
Run Description	unknown, rolling, terse
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49568
Relevant:	2216
Rel-ret:	1717

Recall Level Precision Averages	
Recall	Precision
0.00	0.7836
0.10	0.6631
0.20	0.5907
0.30	0.5384
0.40	0.4790
0.50	0.4162
0.60	0.3446
0.70	0.2597
0.80	0.1851
0.90	0.0851
1.00	0.0290
Average precision over all relevant docs	
non-interpolated	0.3837

Document Level Averages	
	Precision
At 5 docs	0.5440
At 10 docs	0.5080
At 15 docs	0.4907
At 20 docs	0.4560
At 30 docs	0.3920
At 100 docs	0.2146
At 200 docs	0.1339
At 500 docs	0.0636
At 1000 docs	0.0343
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3981



Recall-Precision Curve



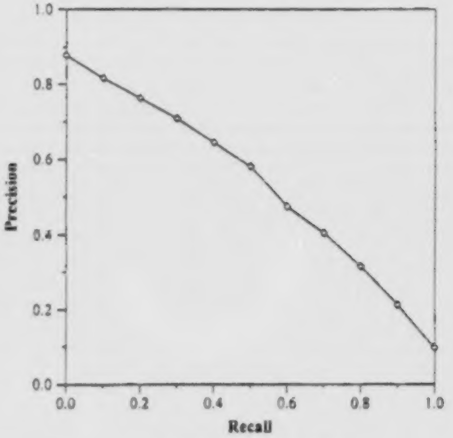
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — Cambridge University

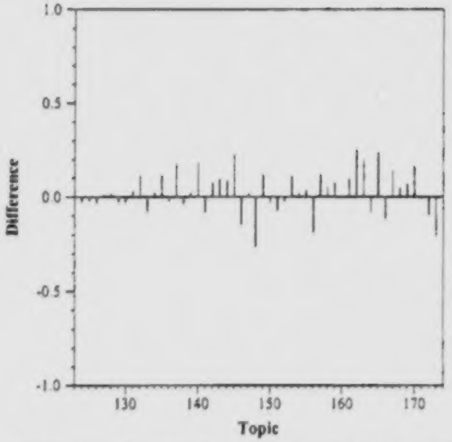
Summary Statistics	
Run Number	cuhtk-r1tk
Run Description	known, fixed, terse
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2123
Rel-ret:	2018

Recall Level Precision Averages	
Recall	Precision
0.00	0.8787
0.10	0.8169
0.20	0.7635
0.30	0.7093
0.40	0.6450
0.50	0.5813
0.60	0.4739
0.70	0.4057
0.80	0.3150
0.90	0.2141
1.00	0.0981
Average precision over all relevant docs	
non-interpolated	0.5268

Document Level Averages	
	Precision
At 5 docs	0.7200
At 10 docs	0.6360
At 15 docs	0.5680
At 20 docs	0.5260
At 30 docs	0.4587
At 100 docs	0.2492
At 200 docs	0.1592
At 500 docs	0.0766
At 1000 docs	0.0404
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4926



Recall-Precision Curve



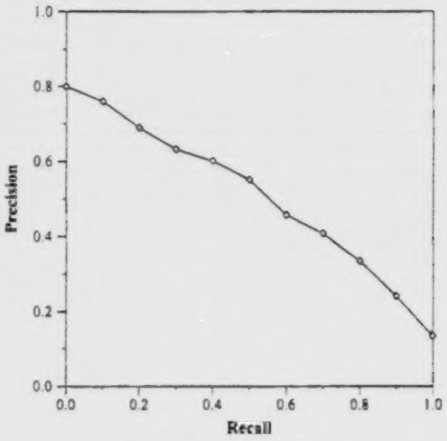
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — Cambridge University

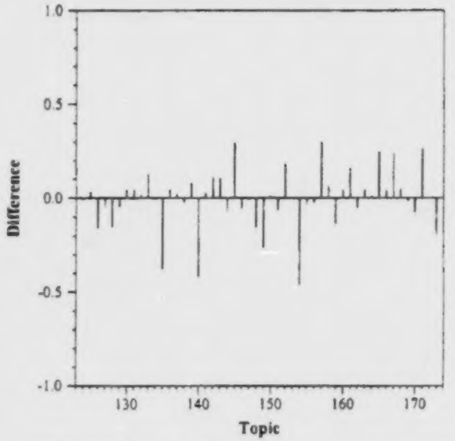
Summary Statistics	
Run Number	cuhtk-r1sk
Run Description	known, fixed, short
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2123
Rel-ret:	1957

Recall Level Precision Averages	
Recall	Precision
0.00	0.8004
0.10	0.7605
0.20	0.6902
0.30	0.6323
0.40	0.6016
0.50	0.5509
0.60	0.4574
0.70	0.4082
0.80	0.3344
0.90	0.2425
1.00	0.1341
Average precision over all relevant docs	
non-interpolated	0.4960

Document Level Averages	
	Precision
At 5 docs	0.6320
At 10 docs	0.5820
At 15 docs	0.5293
At 20 docs	0.4970
At 30 docs	0.4393
At 100 docs	0.2398
At 200 docs	0.1534
At 500 docs	0.0743
At 1000 docs	0.0391
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4705



Recall-Precision Curve



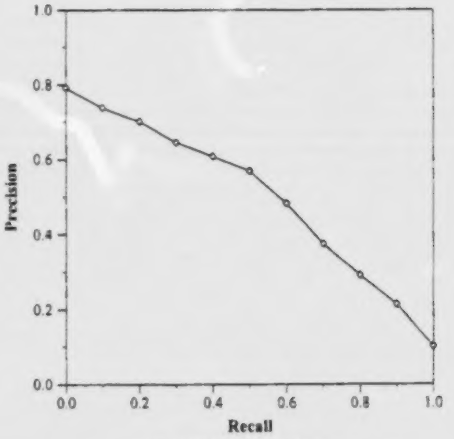
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — Cambridge University

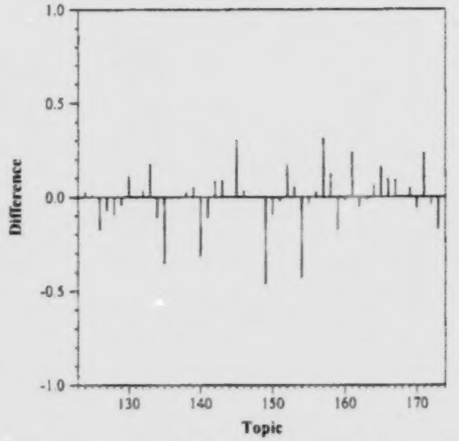
Summary Statistics	
Run Number	cuhtk-s1sk
Run Description	known, fixed, short
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2123
Rel-ret:	1941

Recall Level Precision Averages	
Recall	Precision
0.00	0.7926
0.10	0.7380
0.20	0.7017
0.30	0.6453
0.40	0.6085
0.50	0.5695
0.60	0.4833
0.70	0.3746
0.80	0.2917
0.90	0.2147
1.00	0.1018
Average precision over all relevant docs	
non-interpolated	0.4947

Document Level Averages	
	Precision
At 5 docs	0.6480
At 10 docs	0.6100
At 15 docs	0.5440
At 20 docs	0.4990
At 30 docs	0.4327
At 100 docs	0.2390
At 200 docs	0.1534
At 500 docs	0.0740
At 1000 docs	0.0388
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4783



Recall-Precision Curve



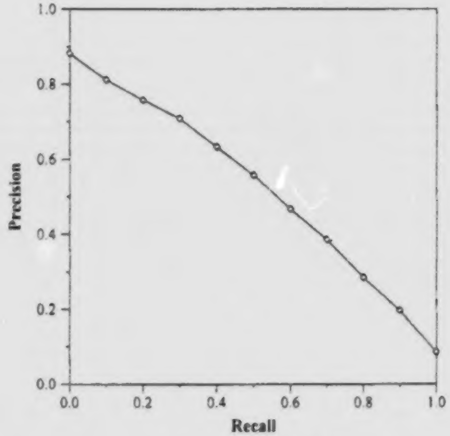
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — Cambridge University

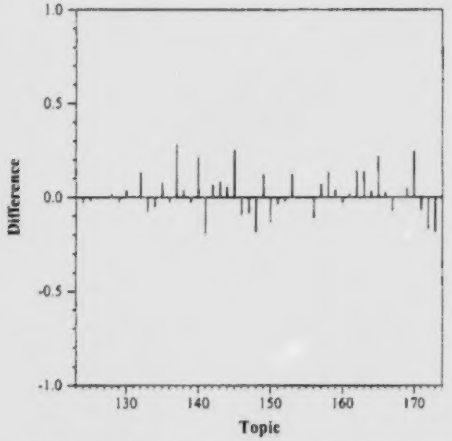
Summary Statistics	
Run Number	cuhtk-s1tk
Run Description	known, fixed, terse
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2123
Rel-ret:	1984

Recall Level Precision Averages	
Recall	Precision
0.00	0.8835
0.10	0.8122
0.20	0.7576
0.30	0.7080
0.40	0.6329
0.50	0.5586
0.60	0.4674
0.70	0.3858
0.80	0.2854
0.90	0.1969
1.00	0.0859
Average precision over all relevant docs	
non-interpolated	0.5194

Document Level Averages	
	Precision
At 5 docs	0.6960
At 10 docs	0.6400
At 15 docs	0.5747
At 20 docs	0.5350
At 30 docs	0.4587
At 100 docs	0.2484
At 200 docs	0.1568
At 500 docs	0.0764
At 1000 docs	0.0397
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5026



Recall-Precision Curve



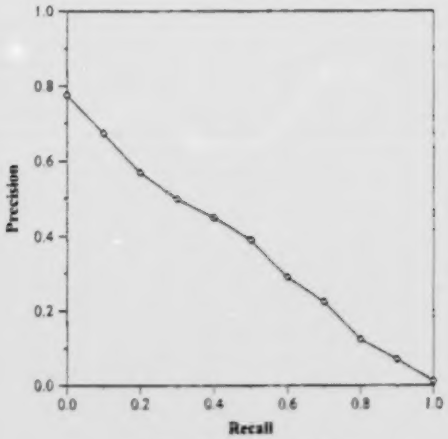
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — U.Sheffield/U.Cambridge/SoftSound/ICSI

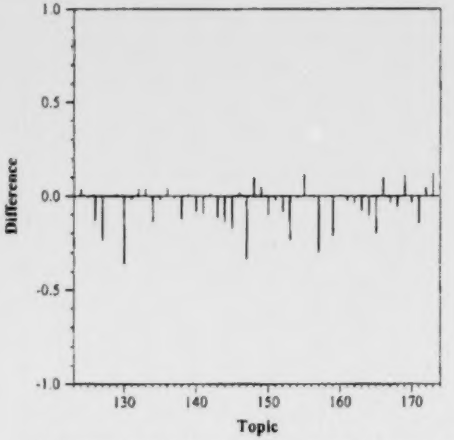
Summary Statistics	
Run Number	shef-crsu-cuhtk1p1u
Run Description	unknown, fixed, short
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49590
Relevant:	2216
Rel-ret:	1648

Recall Level Precision Averages	
Recall	Precision
0.00	0.7763
0.10	0.6731
0.20	0.5695
0.30	0.4982
0.40	0.4484
0.50	0.3899
0.60	0.2900
0.70	0.2239
0.80	0.1245
0.90	0.0708
1.00	0.0116
Average precision over all relevant docs	
non-interpolated	0.3519

Document Level Averages	
	Precision
At 5 docs	0.5520
At 10 docs	0.4860
At 15 docs	0.4533
At 20 docs	0.4270
At 30 docs	0.3627
At 100 docs	0.2062
At 200 docs	0.1290
At 500 docs	0.0622
At 1000 docs	0.0330
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3706



Recall-Precision Curve



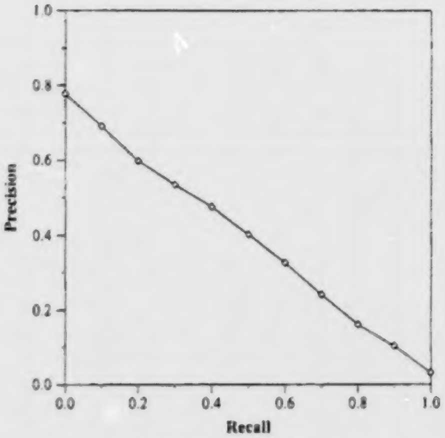
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — U.Sheffield/U.Cambridge/SoftSound/ICSI

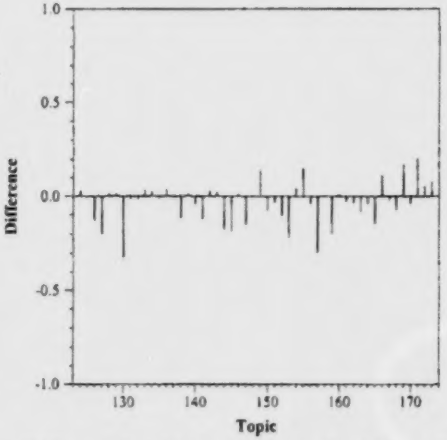
Summary Statistics	
Run Number	shef-crsu-cuhtk1u
Run Description	unknown, fixed, short
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49554
Relevant:	2216
Rel-ret:	1681

Recall Level Precision Averages	
Recall	Precision
0.00	0.7772
0.10	0.6909
0.20	0.5986
0.30	0.5349
0.40	0.4755
0.50	0.4030
0.60	0.3266
0.70	0.2411
0.80	0.1621
0.90	0.1048
1.00	0.0315
Average precision over all relevant docs	
non-interpolated	0.3732

Document Level Averages	
	Precision
At 5 docs	0.5520
At 10 docs	0.4840
At 15 docs	0.4493
At 20 docs	0.4300
At 30 docs	0.3747
At 100 docs	0.2092
At 200 docs	0.1321
At 500 docs	0.0633
At 1000 docs	0.0336
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3878



Recall-Precision Curve



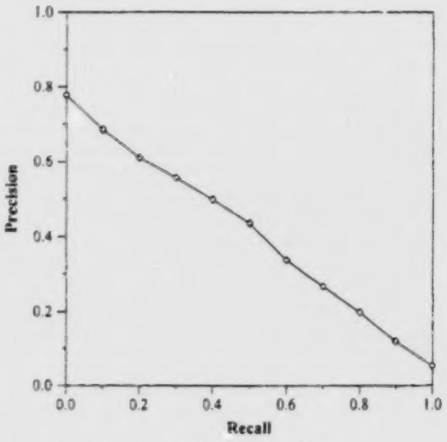
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — U.Sheffield/U.Cambridge/SoftSound/ICSI

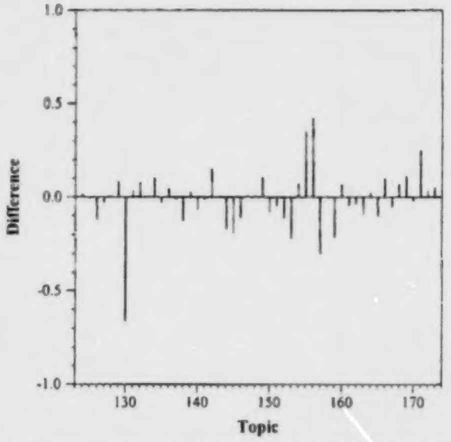
Summary Statistics	
Run Number	shef-crsu-limsi2u
Run Description	unknown, fixed, short
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49604
Relevant:	2216
Rel-ret:	1713

Recall Level Precision Averages	
Recall	Precision
0.00	0.7774
0.10	0.6861
0.20	0.6117
0.30	0.5570
0.40	0.4984
0.50	0.4363
0.60	0.3372
0.70	0.2665
0.80	0.1984
0.90	0.1208
1.00	0.0533
Average precision over all relevant docs	
non-interpolated	0.3952

Document Level Averages	
	Precision
At 5 docs	0.5680
At 10 docs	0.5040
At 15 docs	0.4667
At 20 docs	0.4360
At 30 docs	0.3707
At 100 docs	0.2092
At 200 docs	0.1325
At 500 docs	0.0640
At 1000 docs	0.0343
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4071



Recall-Precision Curve



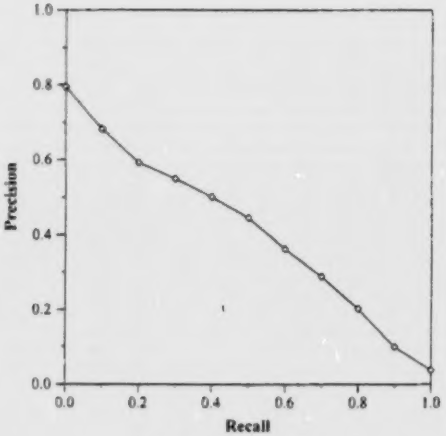
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — U.Sheffield/U.Cambridge/SoftSound/ICSI

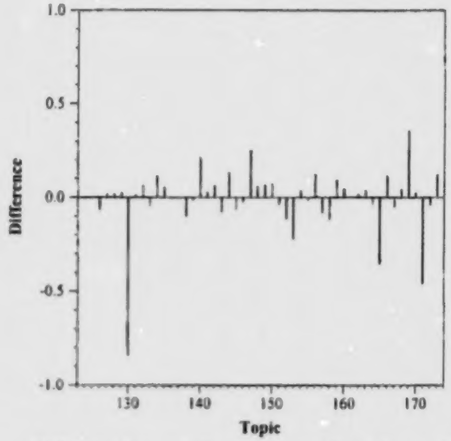
Summary Statistics	
Run Number	shef-crtu-limsi2u
Run Description	unknown, fixed, terse
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49591
Relevant:	2216
Rel-ret:	1743

Recall Level Precision Averages	
Recall	Precision
0.00	0.7949
0.10	0.6827
0.20	0.5924
0.30	0.5487
0.40	0.5014
0.50	0.4462
0.60	0.3621
0.70	0.2872
0.80	0.2029
0.90	0.0994
1.00	0.0371
Average precision over all relevant docs	
non-interpolated	0.3968

Document Level Averages	
	Precision
At 5 docs	0.5600
At 10 docs	0.5020
At 15 docs	0.4653
At 20 docs	0.4400
At 30 docs	0.3840
At 100 docs	0.2144
At 200 docs	0.1353
At 500 docs	0.0644
At 1000 docs	0.0349
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4206



Recall-Precision Curve



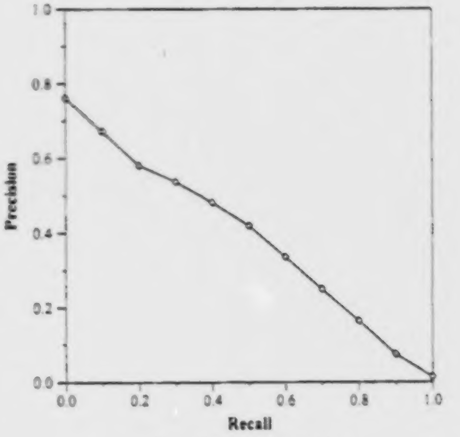
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — U.Sheffield/U.Cambridge/SoftSound/ICSI

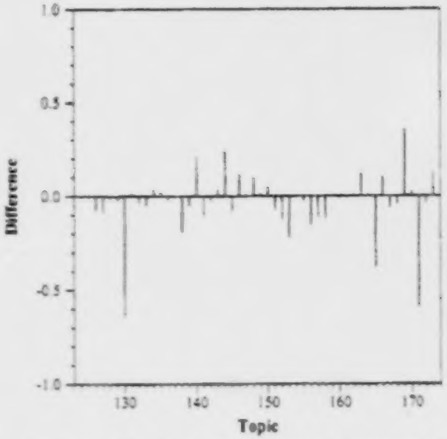
Summary Statistics	
Run Number	shef-crtu-cuhtk1u
Run Description	unknown, fixed, terse
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49425
Relevant:	2216
Rel-ret:	1685

Recall Level Precision Averages	
Recall	Precision
0.00	0.7617
0.10	0.6724
0.20	0.5811
0.30	0.5368
0.40	0.4819
0.50	0.4194
0.60	0.3350
0.70	0.2497
0.80	0.1642
0.90	0.0730
1.00	0.0133
Average precision over all relevant docs	
non-interpolated	0.3727

Document Level Averages	
	Precision
At 5 docs	0.5560
At 10 docs	0.4860
At 15 docs	0.4627
At 20 docs	0.4300
At 30 docs	0.3747
At 100 docs	0.2098
At 200 docs	0.1320
At 500 docs	0.0625
At 1000 docs	0.0337
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3871



Recall-Precision Curve



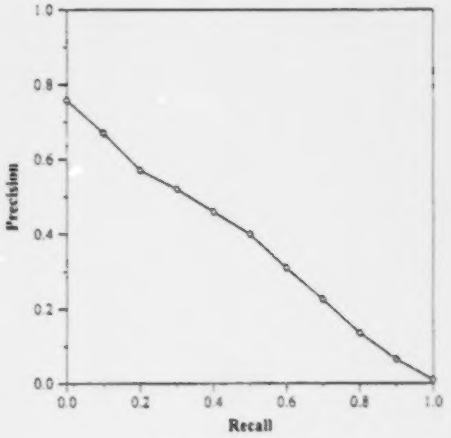
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — U.Sheffield/U.Cambridge/SoftSound/ICSI

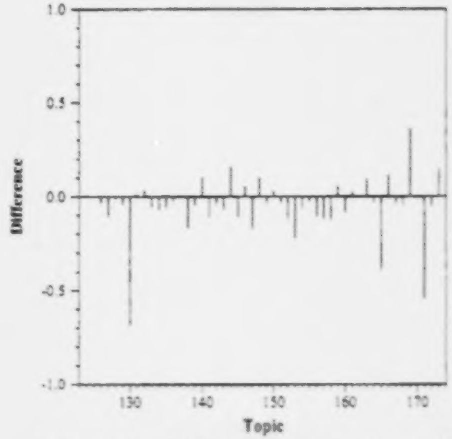
Summary Statistics	
Run Number	shef-crtu-cuhtk1p1u
Run Description	unknown, fixed, terse
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49491
Relevant:	2216
Rel-ret:	1656

Recall Level Precision Averages	
Recall	Precision
0.00	0.7591
0.10	0.6704
0.20	0.5705
0.30	0.5204
0.40	0.4601
0.50	0.3998
0.60	0.3103
0.70	0.2259
0.80	0.1356
0.90	0.0649
1.00	0.0089
Average precision over all relevant docs	
non-interpolated	0.3576

Document Level Averages	
	Precision
At 5 docs	0.5280
At 10 docs	0.4760
At 15 docs	0.4573
At 20 docs	0.4340
At 30 docs	0.3700
At 100 docs	0.2062
At 200 docs	0.1296
At 500 docs	0.0615
At 1000 docs	0.0331
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3763



Recall-Precision Curve



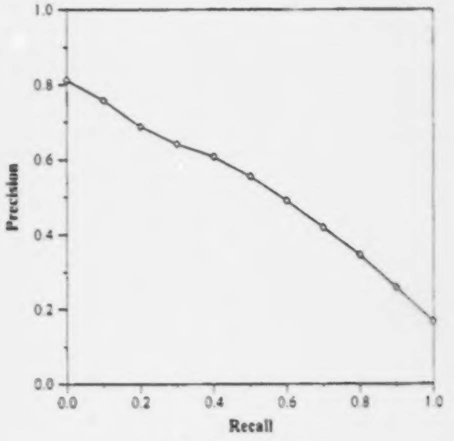
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — U.Sheffield/U.Cambridge/SoftSound/ICSI

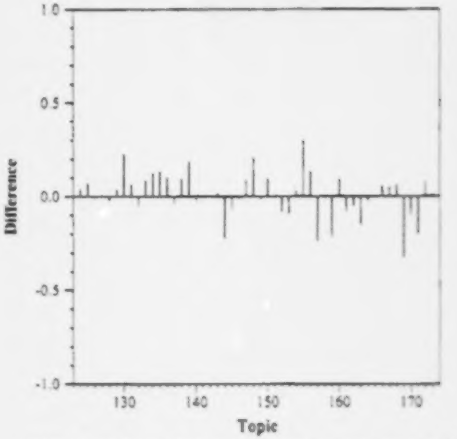
Summary Statistics	
Run Number	shef-r1sk
Run Description	known, fixed, short
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2123
Rel-ret:	1899

Recall Level Precision Averages	
Recall	Precision
0.00	0.8127
0.10	0.7577
0.20	0.6872
0.30	0.6416
0.40	0.6077
0.50	0.5543
0.60	0.4899
0.70	0.4176
0.80	0.3452
0.90	0.2579
1.00	0.1675
Average precision over all relevant docs	
non-interpolated	0.5087

Document Level Averages	
	Precision
At 5 docs	0.6280
At 10 docs	0.5640
At 15 docs	0.5400
At 20 docs	0.4910
At 30 docs	0.4280
At 100 docs	0.2376
At 200 docs	0.1506
At 500 docs	0.0719
At 1000 docs	0.0380
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4894



Recall-Precision Curve

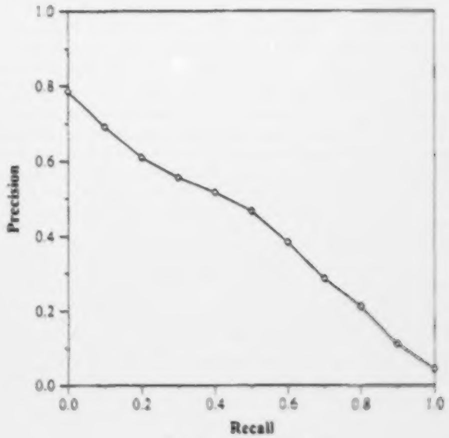


Difference from Median in Average Precision per Topic

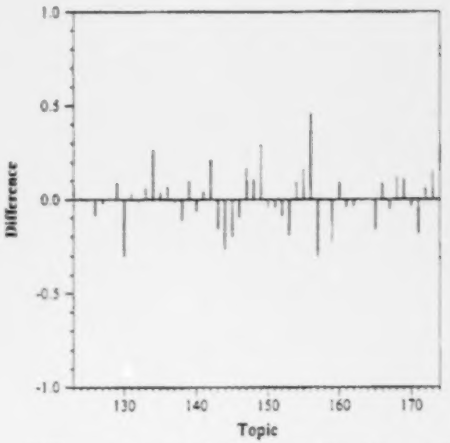
Summary Statistics	
Run Number	shef-r1su
Run Description	unknown, fixed, short
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49452
Relevant:	2216
Rel-ret:	1728

Recall Level Precision Averages	
Recall	Precision
0.00	0.7863
0.10	0.6921
0.20	0.6106
0.30	0.5558
0.40	0.5161
0.50	0.4663
0.60	0.3835
0.70	0.2854
0.80	0.2124
0.90	0.1115
1.00	0.0441
Average precision over all relevant docs	
non-interpolated	0.4093

Document Level Averages	
	Precision
At 5 docs	0.5960
At 10 docs	0.5260
At 15 docs	0.4747
At 20 docs	0.4450
At 30 docs	0.3860
At 100 docs	0.2148
At 200 docs	0.1358
At 500 docs	0.0645
At 1000 docs	0.0346
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4194



Recall-Precision Curve



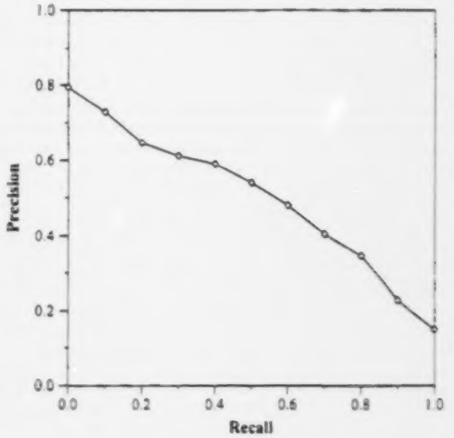
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — U.Sheffield/U.Cambridge/SoftSound/ICSI

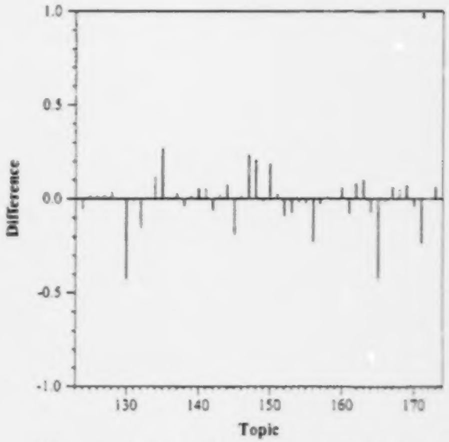
Summary Statistics	
Run Number	shef-r1tk
Run Description	known, fixed, terse
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2123
Rel-ret:	1906

Recall Level Precision Averages	
Recall	Precision
0.00	0.7944
0.10	0.7289
0.20	0.6489
0.30	0.6132
0.40	0.5908
0.50	0.5402
0.60	0.4800
0.70	0.4056
0.80	0.3472
0.90	0.2291
1.00	0.1517
Average precision over all relevant docs	
non-interpolated	0.4920

Document Level Averages	
	Precision
At 5 docs	0.6360
At 10 docs	0.5700
At 15 docs	0.5227
At 20 docs	0.4840
At 30 docs	0.4207
At 100 docs	0.2306
At 200 docs	0.1496
At 500 docs	0.0707
At 1000 docs	0.0381
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4773



Recall-Precision Curve



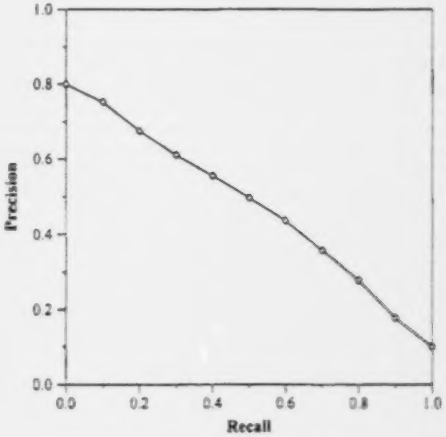
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — U.Sheffield/U.Cambridge/SoftSound/ICSI

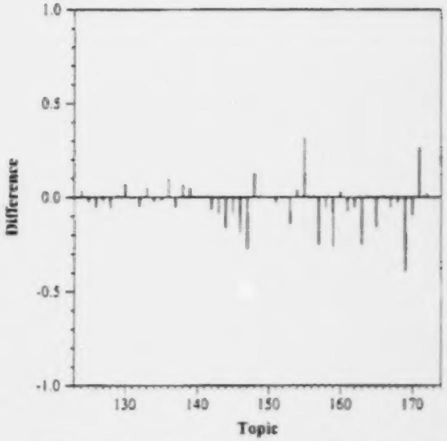
Summary Statistics	
Run Number	shef-s1sk-shef1u
Run Description	known, fixed, short
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2123
Rel-ret:	1799

Recall Level Precision Averages	
Recall	Precision
0.00	0.7999
0.10	0.7533
0.20	0.6760
0.30	0.6106
0.40	0.5554
0.50	0.4985
0.60	0.4367
0.70	0.3574
0.80	0.2786
0.90	0.1781
1.00	0.1000
Average precision over all relevant docs	
non-interpolated	0.4643

Document Level Averages	
	Precision
At 5 docs	0.6280
At 10 docs	0.5580
At 15 docs	0.5080
At 20 docs	0.4660
At 30 docs	0.4007
At 100 docs	0.2158
At 200 docs	0.1412
At 500 docs	0.0674
At 1000 docs	0.0360
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4406



Recall-Precision Curve



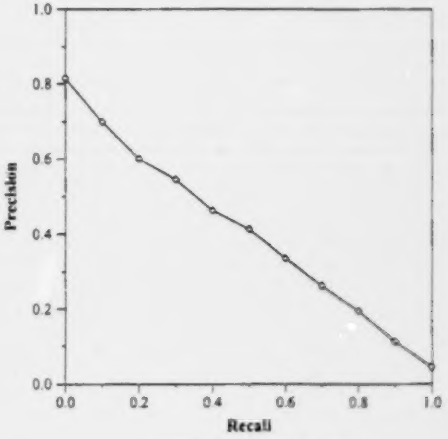
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — U.Sheffield/U.Cambridge/SoftSound/ICSI

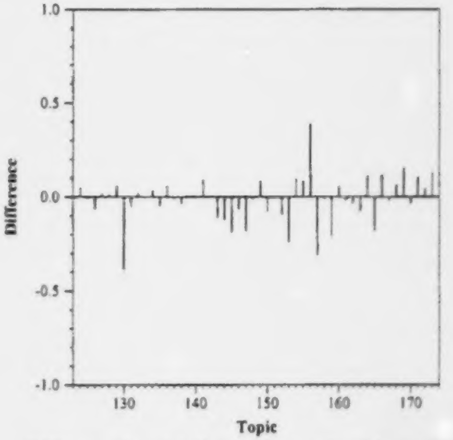
Summary Statistics	
Run Number	shef-s1su-shef1u
Run Description	unknown, fixed, short
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49654
Relevant:	2216
Rel-ret:	1674

Recall Level Precision Averages	
Recall	Precision
0.00	0.8151
0.10	0.6997
0.20	0.6009
0.30	0.5455
0.40	0.4643
0.50	0.4147
0.60	0.3351
0.70	0.2631
0.80	0.1954
0.90	0.1115
1.00	0.0454
Average precision over all relevant docs	
non-interpolated	0.3915

Document Level Averages	
	Precision
At 5 docs	0.5880
At 10 docs	0.5320
At 15 docs	0.4907
At 20 docs	0.4500
At 30 docs	0.3793
At 100 docs	0.2084
At 200 docs	0.1295
At 500 docs	0.0628
At 1000 docs	0.0335
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3990



Recall-Precision Curve



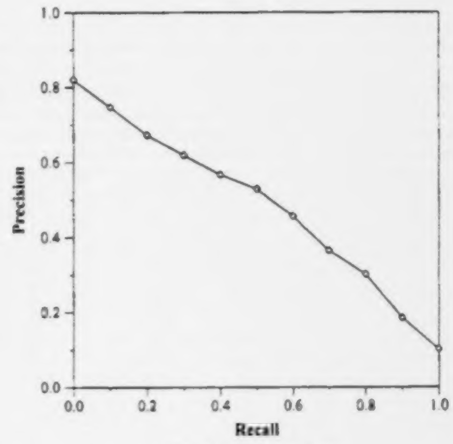
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — U.Sheffield/U.Cambridge/SoftSound/ICSI

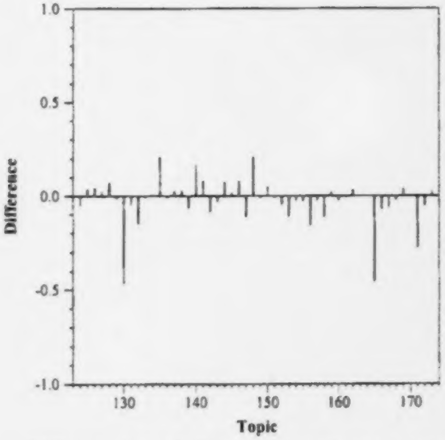
Summary Statistics	
Run Number	shef-s1tk-shef1u
Run Description	known, fixed, terse
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2123
Rel-ret:	1849

Recall Level Precision Averages	
Recall	Precision
0.00	0.8198
0.10	0.7473
0.20	0.6722
0.30	0.6185
0.40	0.5672
0.50	0.5282
0.60	0.4562
0.70	0.3650
0.80	0.3024
0.90	0.1845
1.00	0.1018
Average precision over all relevant docs	
non-interpolated	0.4746

Document Level Averages	
	Precision
At 5 docs	0.6520
At 10 docs	0.5740
At 15 docs	0.5373
At 20 docs	0.4900
At 30 docs	0.4280
At 100 docs	0.2302
At 200 docs	0.1436
At 500 docs	0.0683
At 1000 docs	0.0370
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4630



Recall-Precision Curve



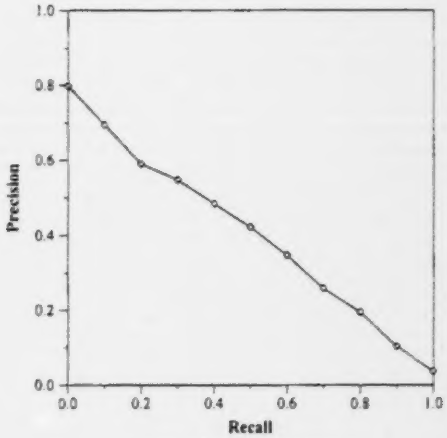
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — U.Sheffield/U.Cambridge/SoftSound/ICSI

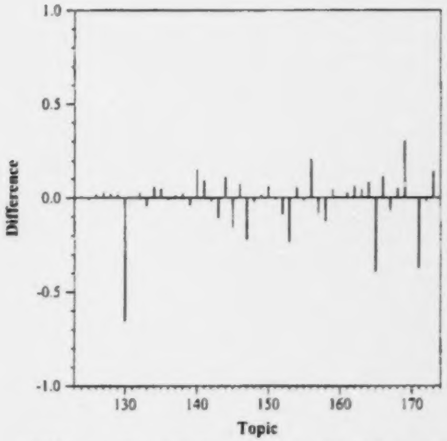
Summary Statistics	
Run Number	shef-s1tu-shef1u
Run Description	unknown, fixed, terse
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49572
Relevant:	2216
Rel-ret:	1708

Recall Level Precision Averages	
Recall	Precision
0.00	0.7987
0.10	0.6956
0.20	0.5897
0.30	0.5473
0.40	0.4844
0.50	0.4235
0.60	0.3464
0.70	0.2599
0.80	0.1950
0.90	0.1033
1.00	0.0370
Average precision over all relevant docs	
non-interpolated	0.3919

Document Level Averages	
	Precision
At 5 docs	0.5760
At 10 docs	0.5240
At 15 docs	0.4920
At 20 docs	0.4670
At 30 docs	0.3907
At 100 docs	0.2136
At 200 docs	0.1334
At 500 docs	0.0632
At 1000 docs	0.0342
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3957



Recall-Precision Curve



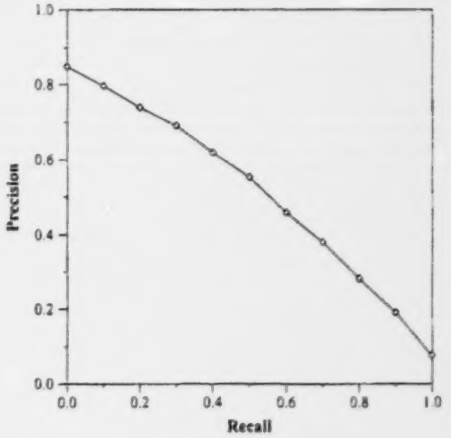
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — Cambridge University

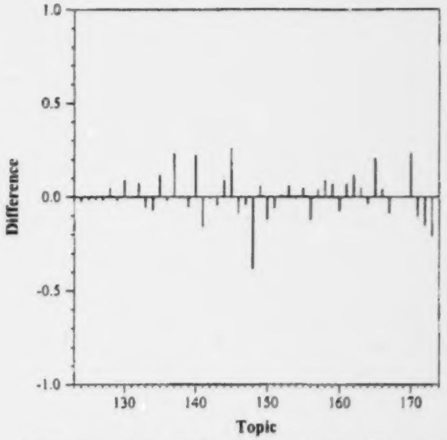
Summary Statistics	
Run Number	cuhtk-b1tk
Run Description	known, fixed, terse
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2123
Rel-ret:	1982

Recall Level Precision Averages	
Recall	Precision
0.00	0.8496
0.10	0.7967
0.20	0.7393
0.30	0.6911
0.40	0.6200
0.50	0.5527
0.60	0.4593
0.70	0.3805
0.80	0.2820
0.90	0.1910
1.00	0.0765
Average precision over all relevant docs	
non-interpolated	0.5044

Document Level Averages	
	Precision
At 5 docs	0.7080
At 10 docs	0.6400
At 15 docs	0.5720
At 20 docs	0.5290
At 30 docs	0.4600
At 100 docs	0.2444
At 200 docs	0.1561
At 500 docs	0.0753
At 1000 docs	0.0396
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4885



Recall-Precision Curve



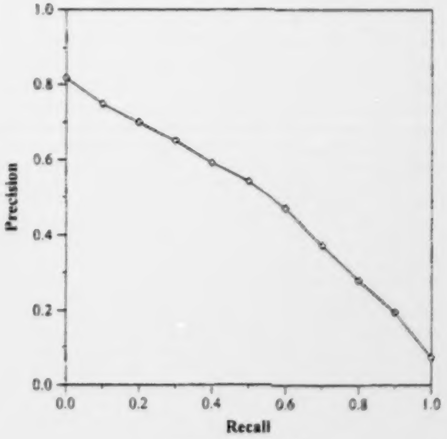
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — Cambridge University

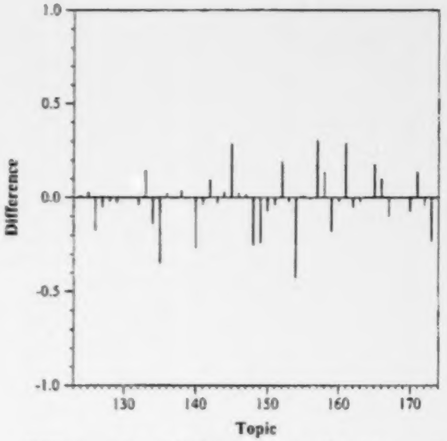
Summary Statistics	
Run Number	cuhtk-b1sk
Run Description	known, fixed, short
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2123
Rel-ret:	1934

Recall Level Precision Averages	
Recall	Precision
0.00	0.8177
0.10	0.7482
0.20	0.7000
0.30	0.6515
0.40	0.5916
0.50	0.5417
0.60	0.4723
0.70	0.3733
0.80	0.2790
0.90	0.1964
1.00	0.0742
Average precision over all relevant docs	
non-interpolated	0.4831

Document Level Averages	
	Precision
At 5 docs	0.6320
At 10 docs	0.6100
At 15 docs	0.5493
At 20 docs	0.5060
At 30 docs	0.4360
At 100 docs	0.2396
At 200 docs	0.1540
At 500 docs	0.0732
At 1000 docs	0.0387
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4738



Recall-Precision Curve



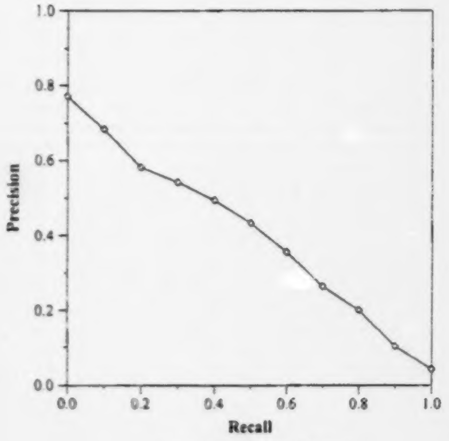
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — U.Sheffield/U.Cambridge/SoftSound/ICSI

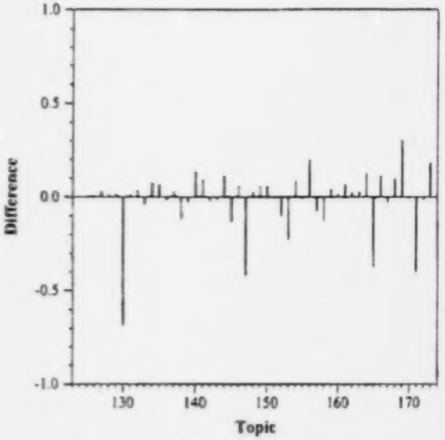
Summary Statistics	
Run Number	shef-s2tu-shef2u
Run Description	unknown, fixed, terse
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49568
Relevant:	2216
Rel-ret:	1734

Recall Level Precision Averages	
Recall	Precision
0.00	0.7712
0.10	0.6846
0.20	0.5817
0.30	0.5425
0.40	0.4958
0.50	0.4352
0.60	0.3573
0.70	0.2653
0.80	0.2026
0.90	0.1036
1.00	0.0428
Average precision over all relevant docs	
non-interpolated	0.3931

Document Level Averages	
	Precision
At 5 docs	0.5680
At 10 docs	0.5200
At 15 docs	0.4960
At 20 docs	0.4570
At 30 docs	0.3933
At 100 docs	0.2160
At 200 docs	0.1354
At 500 docs	0.0640
At 1000 docs	0.0347
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4012



Recall-Precision Curve

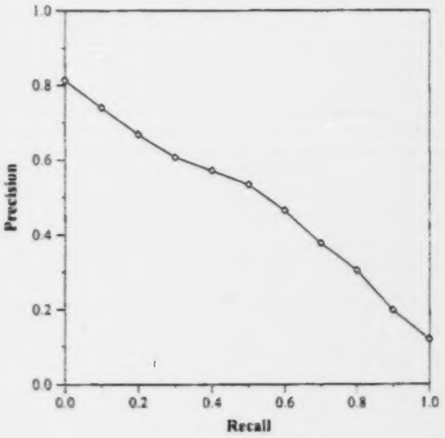


Difference from Median in Average Precision per Topic

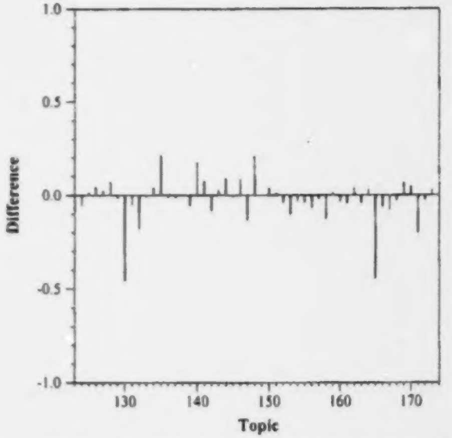
Summary Statistics	
Run Number	shef-s2tk-shef2u
Run Description	known, fixed, terse
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2123
Rel-ret:	1861

Recall Level Precision Averages	
Recall	Precision
0.00	0.8136
0.10	0.7407
0.20	0.6693
0.30	0.6074
0.40	0.5714
0.50	0.5339
0.60	0.4649
0.70	0.3774
0.80	0.3038
0.90	0.1988
1.00	0.1200
Average precision over all relevant docs	
non-interpolated	0.4778

Document Level Averages	
	Precision
At 5 docs	0.6320
At 10 docs	0.5760
At 15 docs	0.5253
At 20 docs	0.4850
At 30 docs	0.4193
At 100 docs	0.2316
At 200 docs	0.1449
At 500 docs	0.0690
At 1000 docs	0.0372
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4629



Recall-Precision Curve



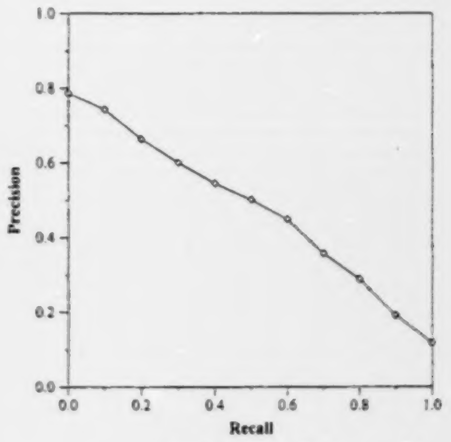
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — U.Sheffield/U.Cambridge/SoftSound/ICSI

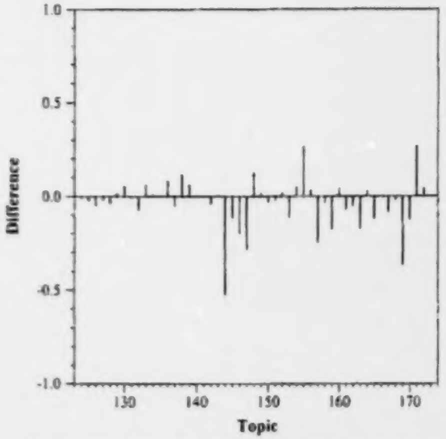
Summary Statistics	
Run Number	shef-s2sk-shef2u
Run Description	known, fixed, short
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2123
Rel-ret:	1820

Recall Level Precision Averages	
Recall	Precision
0.00	0.7868
0.10	0.7437
0.20	0.6645
0.30	0.6007
0.40	0.5463
0.50	0.5012
0.60	0.4490
0.70	0.3569
0.80	0.2894
0.90	0.1920
1.00	0.1183
Average precision over all relevant docs	
non-interpolated	0.4647

Document Level Averages	
	Precision
At 5 docs	0.6120
At 10 docs	0.5520
At 15 docs	0.4987
At 20 docs	0.4670
At 30 docs	0.4060
At 100 docs	0.2208
At 200 docs	0.1434
At 500 docs	0.0684
At 1000 docs	0.0364
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4347



Recall-Precision Curve

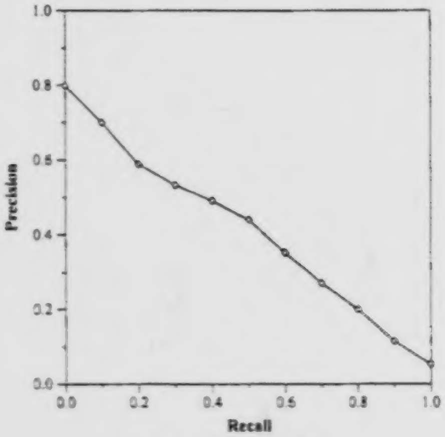


Difference from Median in Average Precision per Topic

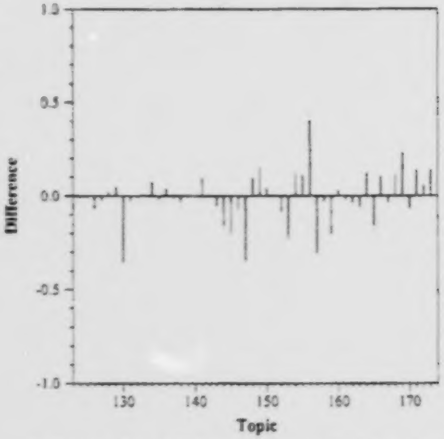
Summary Statistics	
Run Number	shef-s2su-shef2u
Run Description	unknown, fixed, short
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49665
Relevant:	2216
Rel-ret:	1703

Recall Level Precision Averages	
Recall	Precision
0.00	0.7993
0.10	0.7005
0.20	0.5895
0.30	0.5330
0.40	0.4910
0.50	0.4402
0.60	0.3514
0.70	0.2718
0.80	0.2008
0.90	0.1142
1.00	0.0534
Average precision over all relevant docs	
non-interpolated	0.3985

Document Level Averages	
	Precision
At 5 docs	0.6000
At 10 docs	0.5340
At 15 docs	0.4787
At 20 docs	0.4450
At 30 docs	0.3833
At 100 docs	0.2104
At 200 docs	0.1321
At 500 docs	0.0632
At 1000 docs	0.0341
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4099



Recall-Precision Curve



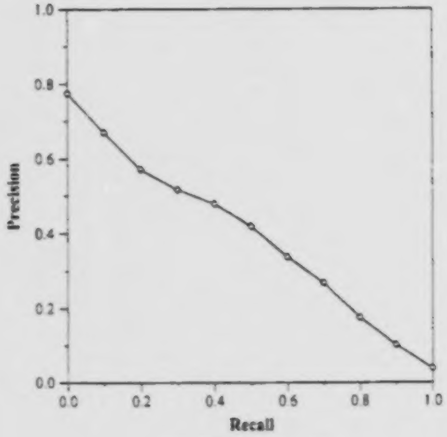
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — U.Sheffield/U.Cambridge/SoftSound/ICSI

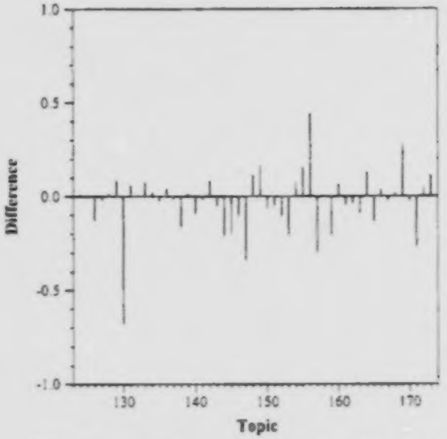
Summary Statistics	
Run Number	shef-crsu-limsi1u
Run Description	unknown, fixed, short
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49628
Relevant:	2216
Rel-ret:	1701

Recall Level Precision Averages	
Recall	Precision
0.00	0.7749
0.10	0.6694
0.20	0.5721
0.30	0.5168
0.40	0.4790
0.50	0.4188
0.60	0.3382
0.70	0.2687
0.80	0.1755
0.90	0.1012
1.00	0.0388
Average precision over all relevant docs	
non-interpolated	0.3766

Document Level Averages	
	Precision
At 5 docs	0.5360
At 10 docs	0.4940
At 15 docs	0.4680
At 20 docs	0.4420
At 30 docs	0.3787
At 100 docs	0.2092
At 200 docs	0.1320
At 500 docs	0.0633
At 1000 docs	0.0340
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4045



Recall-Precision Curve



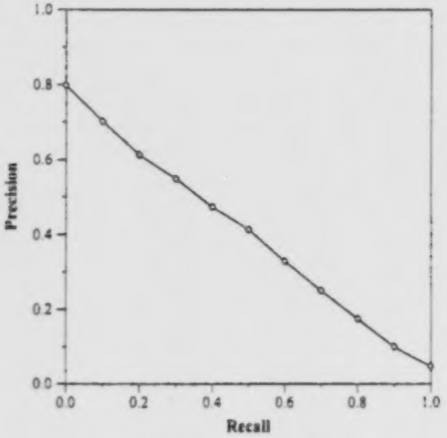
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — U.Sheffield/U.Cambridge/SoftSound/ICSI

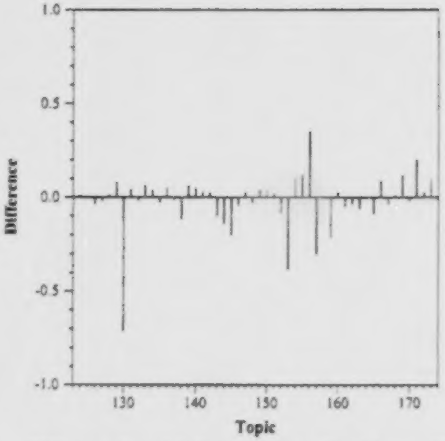
Summary Statistics	
Run Number	shef-b1su
Run Description	unknown, rolling, short
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49666
Relevant:	2216
Rel-ret:	1668

Recall Level Precision Averages	
Recall	Precision
0.00	0.7990
0.10	0.7010
0.20	0.6136
0.30	0.5487
0.40	0.4734
0.50	0.4135
0.60	0.3300
0.70	0.2499
0.80	0.1756
0.90	0.1011
1.00	0.0476
Average precision over all relevant docs	
non-interpolated	0.3872

Document Level Averages	
	Precision
At 5 docs	0.5680
At 10 docs	0.5240
At 15 docs	0.4853
At 20 docs	0.4490
At 30 docs	0.3827
At 100 docs	0.2078
At 200 docs	0.1317
At 500 docs	0.0626
At 1000 docs	0.0334
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4007



Recall-Precision Curve



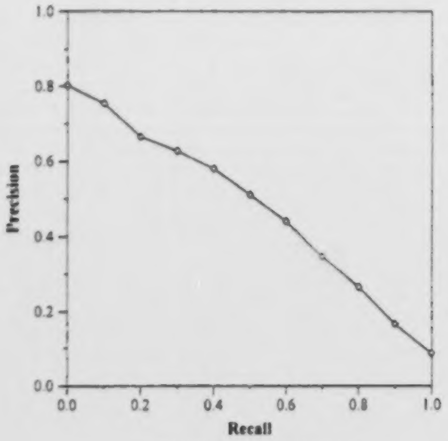
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — U.Sheffield/U.Cambridge/SoftSound/ICSI

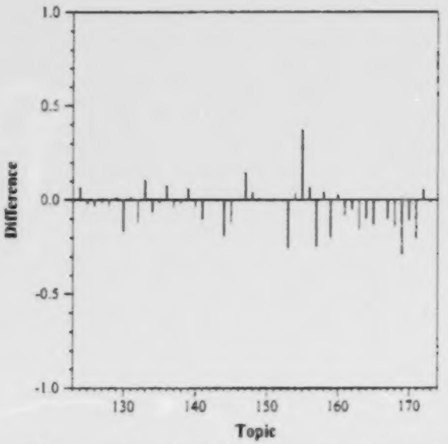
Summary Statistics	
Run Number	shef-b1sk
Run Description	known, rolling, short
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2123
Rel-ret:	1809

Recall Level Precision Averages	
Recall	Precision
0.00	0.8032
0.10	0.7549
0.20	0.6659
0.30	0.6279
0.40	0.5809
0.50	0.5109
0.60	0.4405
0.70	0.3472
0.80	0.2651
0.90	0.1675
1.00	0.0871
Average precision over all relevant docs	
non-interpolated	0.4621

Document Level Averages	
	Precision
At 5 docs	0.6440
At 10 docs	0.5940
At 15 docs	0.5253
At 20 docs	0.4740
At 30 docs	0.4060
At 100 docs	0.2260
At 200 docs	0.1416
At 500 docs	0.0680
At 1000 docs	0.0362
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4465



Recall-Precision Curve



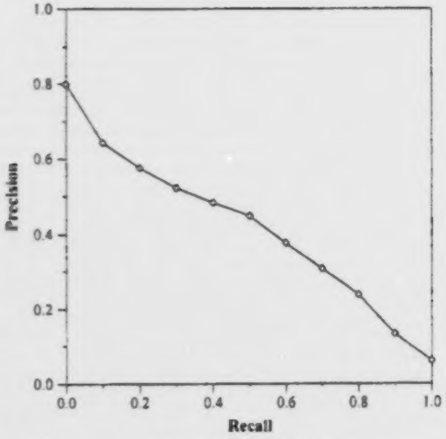
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — LIMSI-CNRS

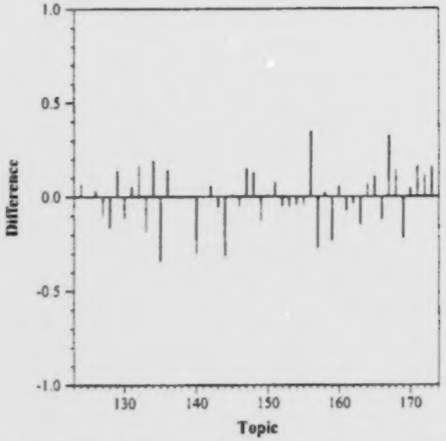
Summary Statistics	
Run Number	limsi-r1su
Run Description	unknown, fixed, short
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49891
Relevant:	2216
Rel-ret:	1793

Recall Level Precision Averages	
Recall	Precision
0.00	0.7994
0.10	0.6442
0.20	0.5762
0.30	0.5219
0.40	0.4831
0.50	0.4483
0.60	0.3772
0.70	0.3076
0.80	0.2384
0.90	0.1348
1.00	0.0609
Average precision over all relevant docs	
non-interpolated	0.4027

Document Level Averages	
	Precision
At 5 docs	0.5560
At 10 docs	0.5020
At 15 docs	0.4733
At 20 docs	0.4300
At 30 docs	0.3667
At 100 docs	0.2126
At 200 docs	0.1353
At 500 docs	0.0660
At 1000 docs	0.0359
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4069



Recall-Precision Curve



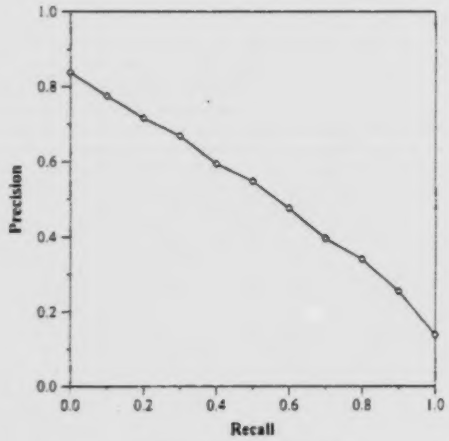
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — LIMSI-CNRS

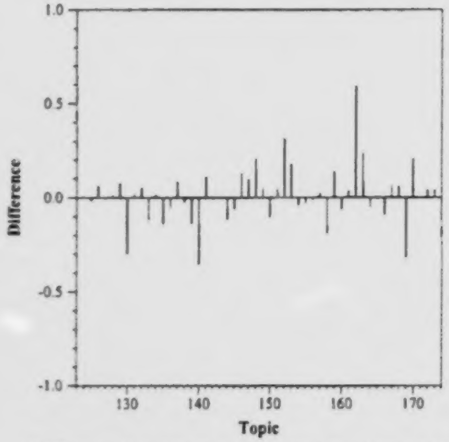
Summary Statistics	
Run Number	limsi-r1tk
Run Description	known, fixed, terse
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2123
Rel-ret:	1998

Recall Level Precision Averages	
Recall	Precision
0.00	0.8383
0.10	0.7754
0.20	0.7164
0.30	0.6691
0.40	0.5950
0.50	0.5476
0.60	0.4757
0.70	0.3961
0.80	0.3411
0.90	0.2550
1.00	0.1394
Average precision over all relevant docs	
non-interpolated	0.5132

Document Level Averages	
	Precision
At 5 docs	0.6640
At 10 docs	0.6080
At 15 docs	0.5573
At 20 docs	0.5130
At 30 docs	0.4553
At 100 docs	0.2488
At 200 docs	0.1608
At 500 docs	0.0756
At 1000 docs	0.0400
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4929



Recall-Precision Curve



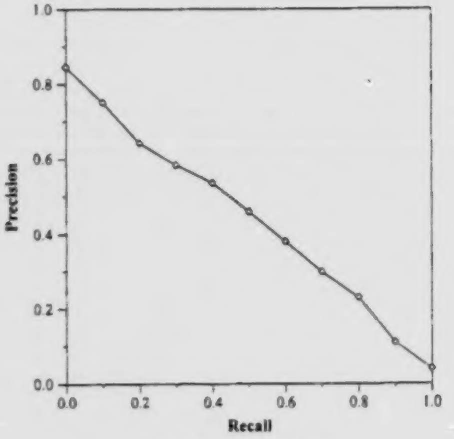
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — LIMSI-CNRS

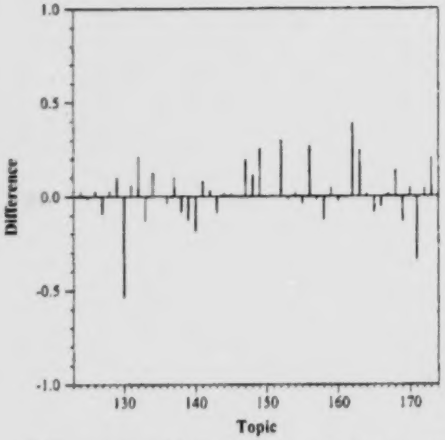
Summary Statistics	
Run Number	limsi-r1tu
Run Description	unknown, fixed, terse
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49936
Relevant:	2216
Rel-ret:	1882

Recall Level Precision Averages	
Recall	Precision
0.00	0.8462
0.10	0.7511
0.20	0.6424
0.30	0.5842
0.40	0.5356
0.50	0.4598
0.60	0.3807
0.70	0.2986
0.80	0.2305
0.90	0.1113
1.00	0.0396
Average precision over all relevant docs	
non-interpolated	0.4283

Document Level Averages	
	Precision
At 5 docs	0.6440
At 10 docs	0.5640
At 15 docs	0.5213
At 20 docs	0.4780
At 30 docs	0.4187
At 100 docs	0.2276
At 200 docs	0.1456
At 500 docs	0.0701
At 1000 docs	0.0376
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4255



Recall-Precision Curve



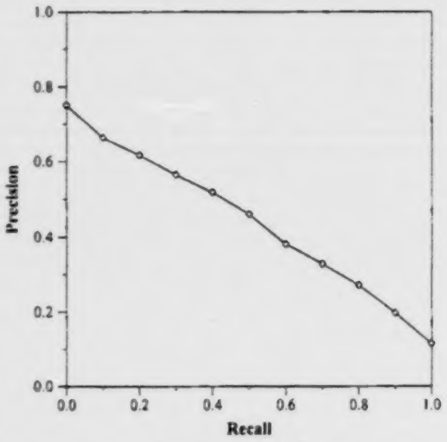
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — LIMSI-CNRS

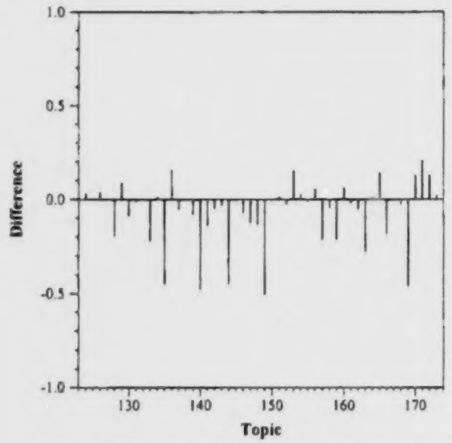
Summary Statistics	
Run Number	limsi-s1sk-limsi2u
Run Description	known, fixed, short
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2123
Rel-ret:	1872

Recall Level Precision Averages	
Recall	Precision
0.00	0.7509
0.10	0.6639
0.20	0.6171
0.30	0.5651
0.40	0.5188
0.50	0.4607
0.60	0.3814
0.70	0.3286
0.80	0.2716
0.90	0.1977
1.00	0.1156
Average precision over all relevant docs	
non-interpolated	0.4327

Document Level Averages	
	Precision
At 5 docs	0.5680
At 10 docs	0.5060
At 15 docs	0.4680
At 20 docs	0.4340
At 30 docs	0.3820
At 100 docs	0.2156
At 200 docs	0.1419
At 500 docs	0.0700
At 1000 docs	0.0374
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4170



Recall-Precision Curve

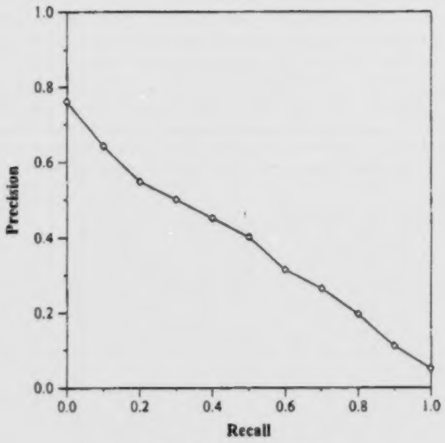


Difference from Median in Average Precision per Topic

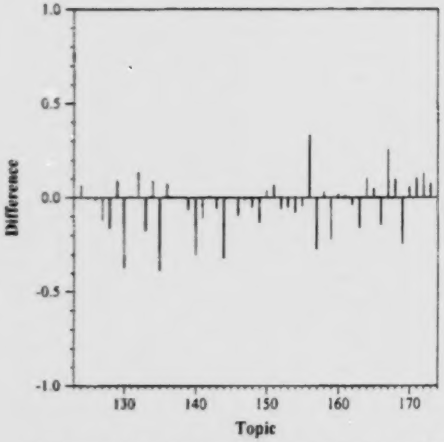
Summary Statistics	
Run Number	limsi-s1su-limsi2u
Run Description	unknown, fixed, short
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49936
Relevant:	2216
Rel-ret:	1760

Recall Level Precision Averages	
Recall	Precision
0.00	0.7613
0.10	0.6441
0.20	0.5487
0.30	0.5011
0.40	0.4522
0.50	0.4032
0.60	0.3148
0.70	0.2655
0.80	0.1965
0.90	0.1122
1.00	0.0510
Average precision over all relevant docs	
non-interpolated	0.3706

Document Level Averages	
	Precision
At 5 docs	0.5240
At 10 docs	0.4820
At 15 docs	0.4507
At 20 docs	0.4120
At 30 docs	0.3520
At 100 docs	0.2066
At 200 docs	0.1308
At 500 docs	0.0649
At 1000 docs	0.0352
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3788



Recall-Precision Curve

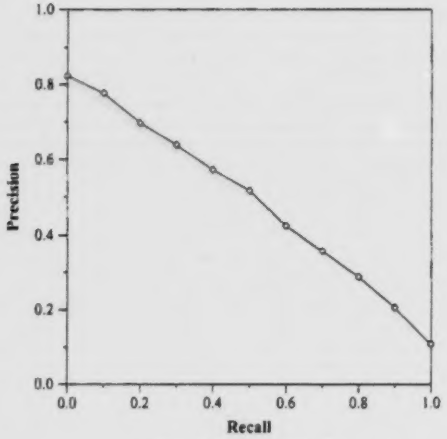


Difference from Median in Average Precision per Topic

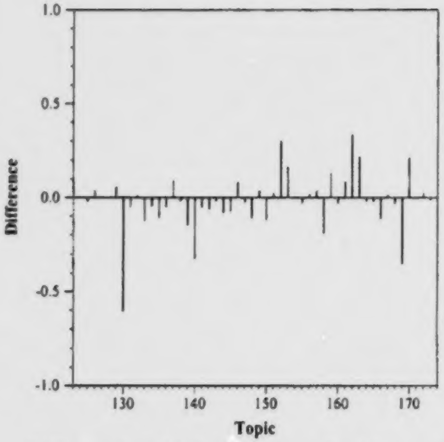
Summary Statistics	
Run Number	limsi-s1tk-limsi2u
Run Description	known, fixed, terse
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2123
Rel-ret:	1973

Recall Level Precision Averages	
Recall	Precision
0.00	0.8237
0.10	0.7765
0.20	0.6976
0.30	0.6395
0.40	0.5728
0.50	0.5163
0.60	0.4244
0.70	0.3572
0.80	0.2877
0.90	0.2051
1.00	0.1090
Average precision over all relevant docs	
non-interpolated	0.4812

Document Level Averages	
	Precision
At 5 docs	0.6480
At 10 docs	0.5880
At 15 docs	0.5453
At 20 docs	0.4960
At 30 docs	0.4380
At 100 docs	0.2432
At 200 docs	0.1573
At 500 docs	0.0754
At 1000 docs	0.0395
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4598



Recall-Precision Curve



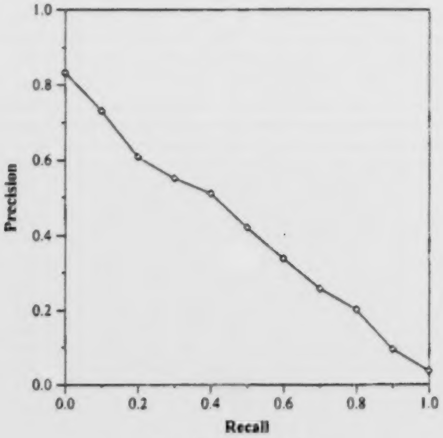
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — LIMSI-CNRS

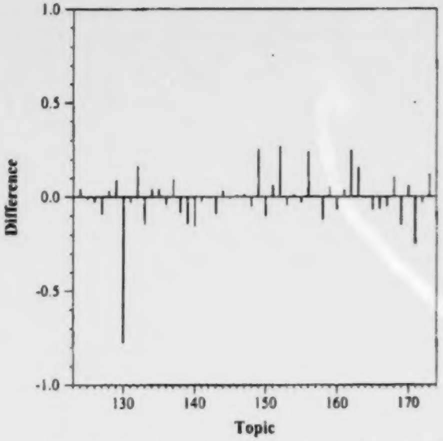
Summary Statistics	
Run Number	limsi-s1tu-limsi2u
Run Description	unknown, fixed, terse
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49963
Relevant:	2216
Rel-ret:	1868

Recall Level Precision Averages	
Recall	Precision
0.00	0.8334
0.10	0.7308
0.20	0.6086
0.30	0.5500
0.40	0.5100
0.50	0.4217
0.60	0.3377
0.70	0.2572
0.80	0.2023
0.90	0.0952
1.00	0.0358
Average precision over all relevant docs	
non-interpolated	0.3982

Document Level Averages	
	Precision
At 5 docs	0.5880
At 10 docs	0.5360
At 15 docs	0.4960
At 20 docs	0.4670
At 30 docs	0.4000
At 100 docs	0.2212
At 200 docs	0.1442
At 500 docs	0.0692
At 1000 docs	0.0374
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4099



Recall-Precision Curve

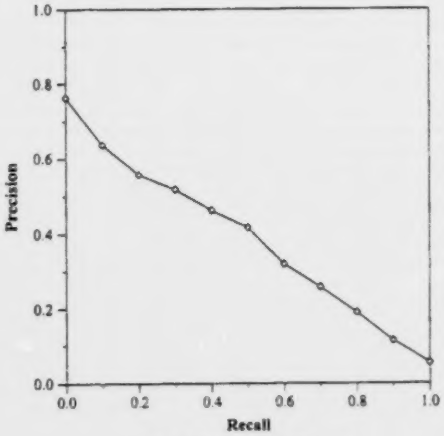


Difference from Median in Average Precision per Topic

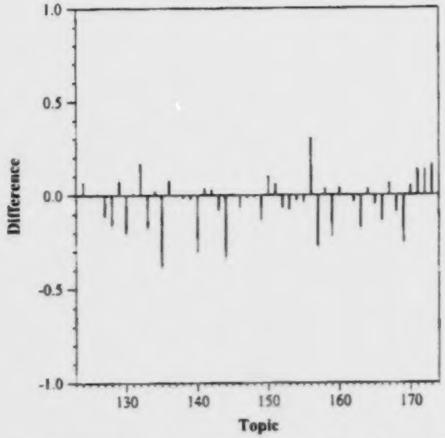
Summary Statistics	
Run Number	limsi-b1su
Run Description	unknown, rolling, short
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49958
Relevant:	2216
Rel-ret:	1729

Recall Level Precision Averages	
Recall	Precision
0.00	0.7647
0.10	0.6370
0.20	0.5565
0.30	0.5190
0.40	0.4635
0.50	0.4169
0.60	0.3182
0.70	0.2576
0.80	0.1908
0.90	0.1136
1.00	0.0543
Average precision over all relevant docs	
non-interpolated	0.3712

Document Level Averages	
	Precision
At 5 docs	0.5240
At 10 docs	0.4880
At 15 docs	0.4480
At 20 docs	0.4160
At 30 docs	0.3607
At 100 docs	0.2038
At 200 docs	0.1273
At 500 docs	0.0638
At 1000 docs	0.0346
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3801



Recall-Precision Curve



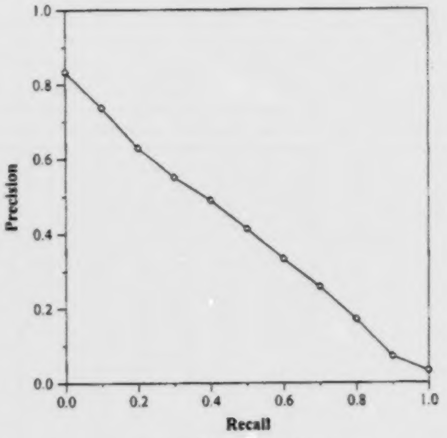
Difference from Median in Average Precision per Topic

Spoken document retrieval track results — LIMSI-CNRS

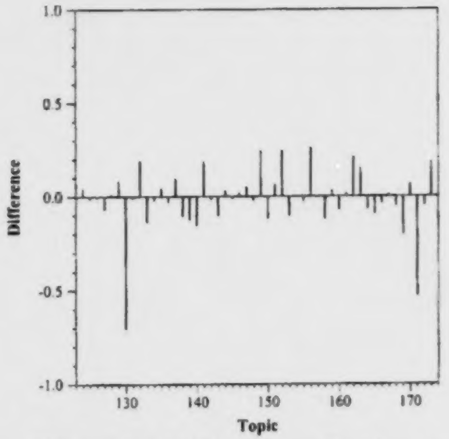
Summary Statistics	
Run Number	limsi-b1tu
Run Description	unknown, rolling, terse
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2216
Rel-ret:	1829

Recall Level Precision Averages	
Recall	Precision
0.00	0.8336
0.10	0.7381
0.20	0.6288
0.30	0.5514
0.40	0.4902
0.50	0.4125
0.60	0.3326
0.70	0.2587
0.80	0.1700
0.90	0.0701
1.00	0.0317
Average precision over all relevant docs	
non-interpolated	0.3922

Document Level Averages	
	Precision
At 5 docs	0.6040
At 10 docs	0.5520
At 15 docs	0.5080
At 20 docs	0.4670
At 30 docs	0.3953
At 100 docs	0.2216
At 200 docs	0.1408
At 500 docs	0.0674
At 1000 docs	0.0366
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4000



Recall-Precision Curve



Difference from Median in Average Precision per Topic

TREC-9 Web Results

Publications home NIST Special Publication XXX- XXX

Help

- NEnmLpas
- NEnmLsa
- NEnm
- NEtm
- NENRtm
- NENRtmLpas
- iit00m
- PuShortAuth
- PuShortWAuth
- PuShortBase
- PuLongAuth
- PuLongWAuth
- PuLongBase
- iit00td
- iit00tde
- iit00t
- CWI0001
- CWI0000
- CWI0002

- CWI0010
- hum9tdn
- pir0Wttd
- pir0Watd
- pir0Wtd2
- pir0Wt1
- hum9te
- hum9tde
- hum9td4
- rmitWFGweb
- rmitNFGweb
- rmitWFLweb
- rmitNFLweb
- jscbt9wll1
- jscbt9wls1
- jscbt9wcs1
- jscbt9wcl1
- pir0WTTD
- Sab9web1
- Sab9web2
- Sab9web3
- Sab9web4
- jscbt9wll2
- jscbt9wls2
- NRKlm

- NRKprf20
- NRKse20
- NRKse10
- ric9dpx
- ric9dpn
- acsys9mw0
- Mer9Wtnd
- Mer9WtdMr
- Mer9Wt0
- Mer9Wt1
- dcu00ca
- dcu00la
- dcu00lc
- dcu00lb
- tnout9t2lk50
- tnout9t2lc50
- tnout9t2lc10
- Flab9atN
- Flab9atdN
- Flab9atdnN
- tnout9t2
- tnout9f1
- uwmt9w10g0
- uwmt9w10g1
- Sab9web5

- xvsmmain
- ric9dsx
- xvsmtdn
- apl9tdn
- apl9t
- uwmt9w10g2
- apl9lt
- uwmt9w10g3
- apl9ltdn
- xvsmtitle
- ric9dpxL
- xvsmman
- Flab9atd2N
- Scai9Web1
- Scai9Web3
- Scai9Web2
- Scai9Web4
- att0010gbe
- att0010gb
- att0010gbt
- att0010gbl
- att0010glf
- att0010glv
- uwmt9w10g4
- uwmt9w10g5

- UCCS3
- UCCS4
- UCCS1
- UCCS2
- ric9tpx
- iswtd
- iswtdn
- iswt
- isnnwt
- apl9all
- apl9td

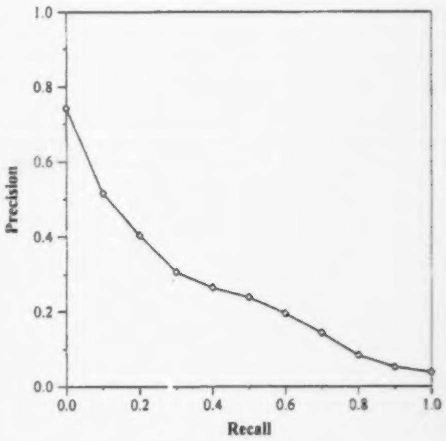
Last updated: Tuesday, 28-Aug-01 09:51:10
Date created: Tuesday, 28-Aug-01
trec@nist.gov

Main web track results — Universite de Neuchatel

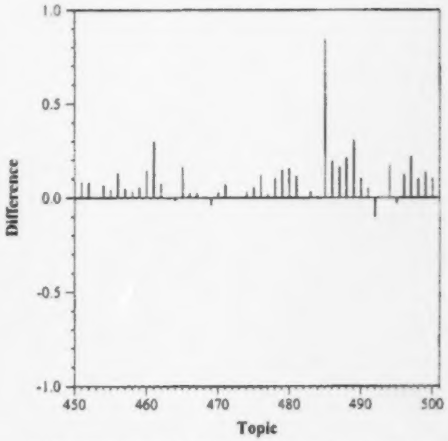
Summary Statistics	
Run Number	NEnmLpas
Run Description	Automatic, content-link, title+desc+narr
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1704

Recall Level Precision Averages	
Recall	Precision
0.00	0.7428
0.10	0.5160
0.20	0.4035
0.30	0.3076
0.40	0.2651
0.50	0.2397
0.60	0.1954
0.70	0.1433
0.80	0.0847
0.90	0.0535
1.00	0.0389
Average precision over all relevant docs	
non-interpolated	0.2488

Document Level Averages	
	Precision
At 5 docs	0.4360
At 10 docs	0.3400
At 15 docs	0.2933
At 20 docs	0.2700
At 30 docs	0.2413
At 100 docs	0.1536
At 200 docs	0.1067
At 500 docs	0.0586
At 1000 docs	0.0341
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2832



Recall-Precision Curve

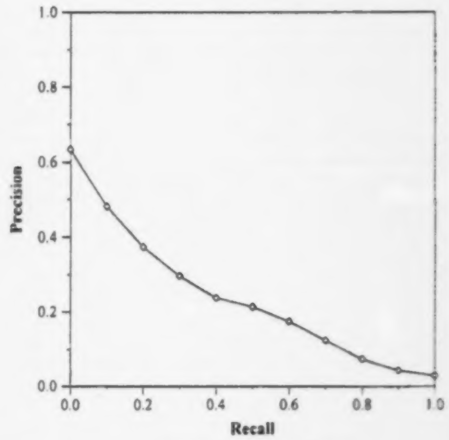


Difference from Median in Average Precision per Topic

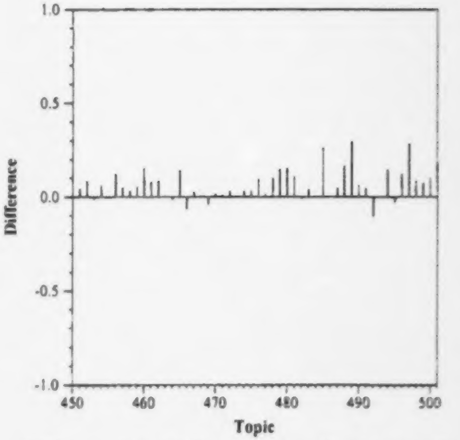
Summary Statistics	
Run Number	NEnmLsa
Run Description	Automatic, content-link, title+desc+narr
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1705

Recall Level Precision Averages	
Recall	Precision
0.00	0.6340
0.10	0.4806
0.20	0.3740
0.30	0.2961
0.40	0.2372
0.50	0.2126
0.60	0.1740
0.70	0.1241
0.80	0.0747
0.90	0.0438
1.00	0.0294
Average precision over all relevant docs	
non-interpolated	0.2185

Document Level Averages	
	Precision
At 5 docs	0.4080
At 10 docs	0.3320
At 15 docs	0.2960
At 20 docs	0.2700
At 30 docs	0.2413
At 100 docs	0.1536
At 200 docs	0.1070
At 500 docs	0.0585
At 1000 docs	0.0341
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2427



Recall-Precision Curve



Difference from Median in Average Precision per Topic

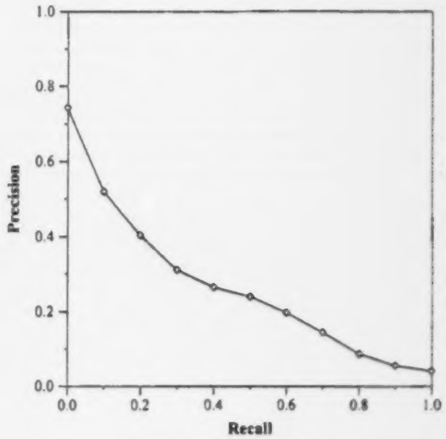
Main web track results — Universite de Neuchatel

Summary Statistics	
Run Number	NEnm
Run Description	Automatic, content-only, title+desc+narr
Number of Topics	50

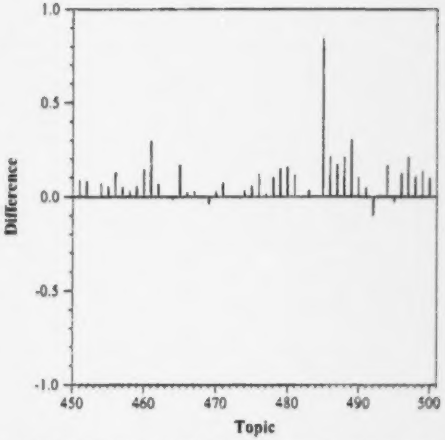
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1704

Recall Level Precision Averages	
Recall	Precision
0.00	0.7430
0.10	0.5185
0.20	0.4039
0.30	0.3106
0.40	0.2648
0.50	0.2395
0.60	0.1975
0.70	0.1449
0.80	0.0867
0.90	0.0554
1.00	0.0408
Average precision over all relevant docs	
non-interpolated	0.2499

Document Level Averages	
	Precision
At 5 docs	0.4400
At 10 docs	0.3420
At 15 docs	0.3027
At 20 docs	0.2710
At 30 docs	0.2407
At 100 docs	0.1536
At 200 docs	0.1068
At 500 docs	0.0586
At 1000 docs	0.0341
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2871



Recall-Precision Curve



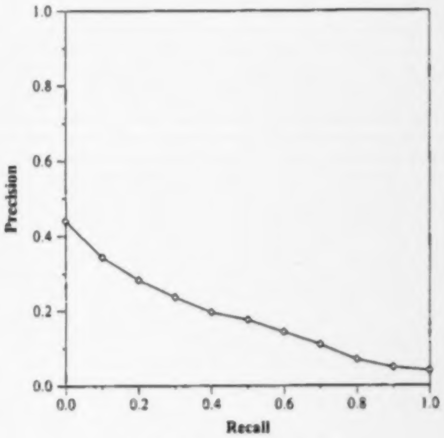
Difference from Median in Average Precision per Topic

Main web track results — Universite de Neuchatel

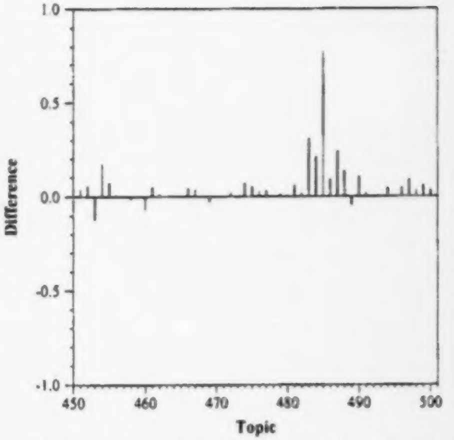
Summary Statistics	
Run Number	NEtm
Run Description	Automatic, content-only, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1323

Recall Level Precision Averages	
Recall	Precision
0.00	0.4394
0.10	0.3433
0.20	0.2830
0.30	0.2377
0.40	0.1965
0.50	0.1762
0.60	0.1439
0.70	0.1098
0.80	0.0693
0.90	0.0484
1.00	0.0398
Average precision over all relevant docs	
non-interpolated	0.1754

Document Level Averages	
	Precision
At 5 docs	0.2120
At 10 docs	0.2120
At 15 docs	0.1987
At 20 docs	0.1860
At 30 docs	0.1600
At 100 docs	0.1088
At 200 docs	0.0786
At 500 docs	0.0443
At 1000 docs	0.0265
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2033



Recall-Precision Curve

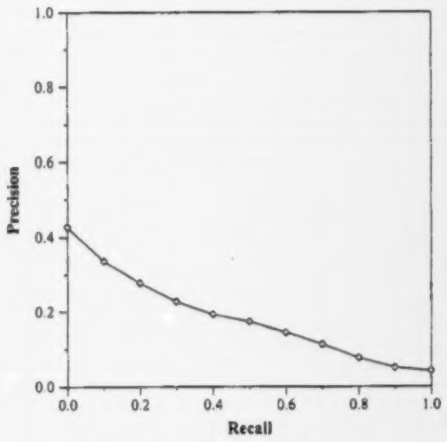


Difference from Median in Average Precision per Topic

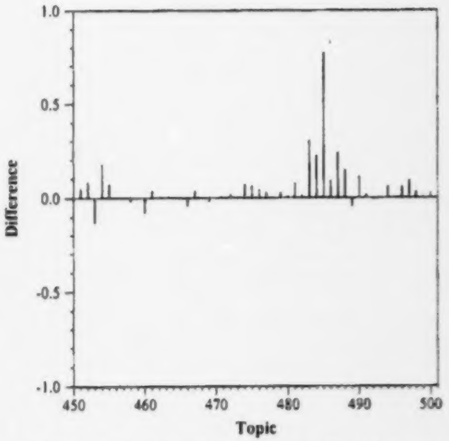
Summary Statistics	
Run Number	NENRtm
Run Description	Automatic, content-only, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1340

Recall Level Precision Averages	
Recall	Precision
0.00	0.4265
0.10	0.3357
0.20	0.2785
0.30	0.2290
0.40	0.1938
0.50	0.1737
0.60	0.1445
0.70	0.1131
0.80	0.0772
0.90	0.0510
1.00	0.0415
Average precision over all relevant docs	
non-interpolated	0.1743

Document Level Averages	
	Precision
At 5 docs	0.2080
At 10 docs	0.2080
At 15 docs	0.2027
At 20 docs	0.1860
At 30 docs	0.1640
At 100 docs	0.1112
At 200 docs	0.0812
At 500 docs	0.0449
At 1000 docs	0.0268
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2030



Recall-Precision Curve



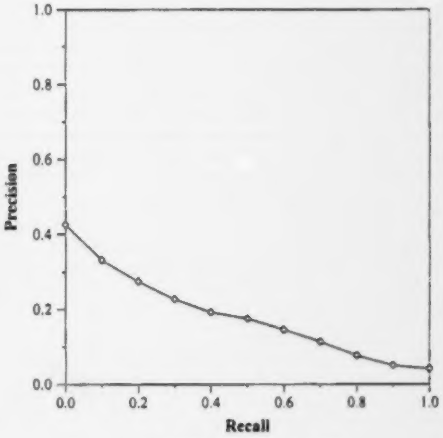
Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	NENRtmLpas
Run Description	Automatic, content-link, title only
Number of Topics	50

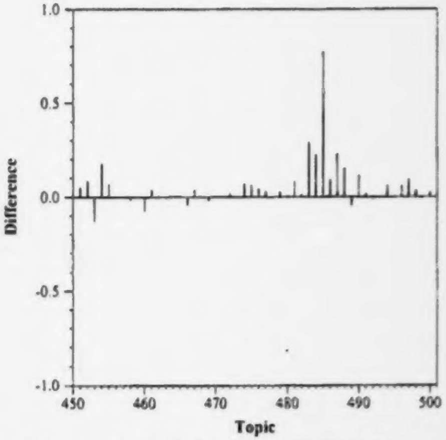
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1340

Recall Level Precision Averages	
Recall	Precision
0.00	0.4258
0.10	0.3325
0.20	0.2763
0.30	0.2275
0.40	0.1928
0.50	0.1751
0.60	0.1455
0.70	0.1131
0.80	0.0772
0.90	0.0510
1.00	0.0414
Average precision over all relevant docs	
non-interpolated	0.1736

Document Level Averages	
	Precision
At 5 docs	0.2080
At 10 docs	0.2080
At 15 docs	0.2000
At 20 docs	0.1840
At 30 docs	0.1640
At 100 docs	0.1112
At 200 docs	0.0812
At 500 docs	0.0449
At 1000 docs	0.0268
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2039



Recall-Precision Curve



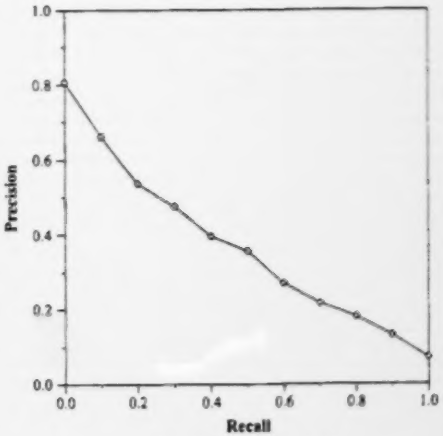
Difference from Median in Average Precision per Topic

Main web track results — IIT/AAT/NCR

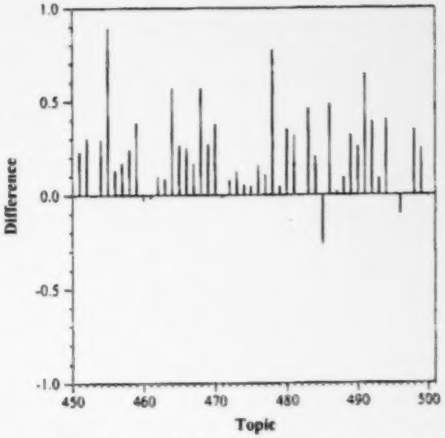
Summary Statistics	
Run Number	iit00m
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	25956
Relevant:	2617
Rel-ret:	1617

Recall Level Precision Averages	
Recall	Precision
0.00	0.8073
0.10	0.6606
0.20	0.5366
0.30	0.4753
0.40	0.3963
0.50	0.3570
0.60	0.2702
0.70	0.2160
0.80	0.1806
0.90	0.1303
1.00	0.0717
Average precision over all relevant docs	
non-interpolated	0.3519

Document Level Averages	
	Precision
At 5 docs	0.5920
At 10 docs	0.5180
At 15 docs	0.4547
At 20 docs	0.4090
At 30 docs	0.3440
At 100 docs	0.1922
At 200 docs	0.1207
At 500 docs	0.0586
At 1000 docs	0.0323
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3708



Recall-Precision Curve



Difference from Median in Average Precision per Topic

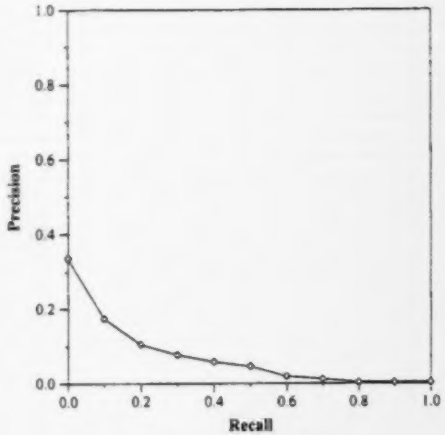
Main web track results — University of Padova

Summary Statistics	
Run Number	PuShortAuth
Run Description	Automatic, content-link, title only
Number of Topics	50

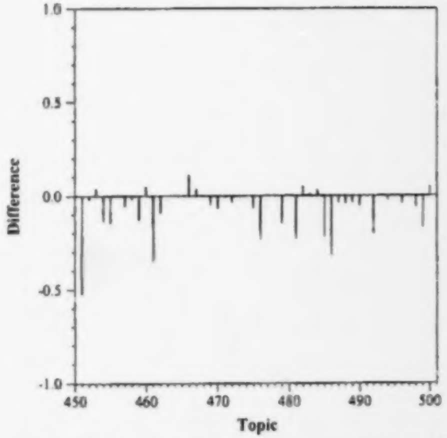
Total number of documents over all topics	
Retrieved:	42538
Relevant:	2617
Rel-ret:	713

Recall Level Precision Averages	
Recall	Precision
0.00	0.3368
0.10	0.1738
0.20	0.1055
0.30	0.0774
0.40	0.0588
0.50	0.0470
0.60	0.0197
0.70	0.0120
0.80	0.0033
0.90	0.0033
1.00	0.0033
Average precision over all relevant docs	
non-interpolated	0.0591

Document Level Averages	
	Precision
At 5 docs	0.1600
At 10 docs	0.1360
At 15 docs	0.1187
At 20 docs	0.1050
At 30 docs	0.0953
At 100 docs	0.0612
At 200 docs	0.0399
At 500 docs	0.0239
At 1000 docs	0.0143
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0985



Recall-Precision Curve



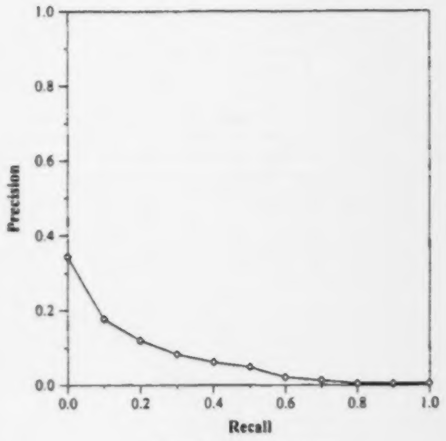
Difference from Median in Average Precision per Topic

Main web track results — University of Padova

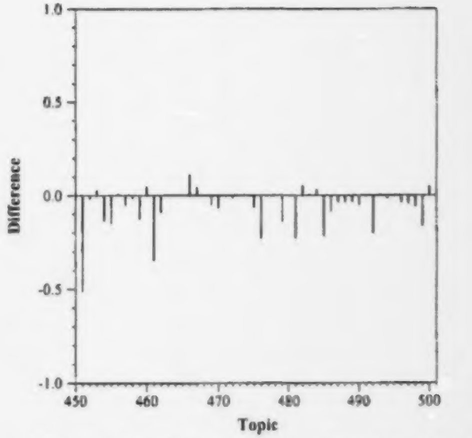
Summary Statistics	
Run Number	PuShortWAuth
Run Description	Automatic, content-link, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	42538
Relevant:	2617
Rel-ret:	713

Recall Level Precision Averages	
Recall	Precision
0.00	0.3447
0.10	0.1763
0.20	0.1199
0.30	0.0830
0.40	0.0614
0.50	0.0480
0.60	0.0210
0.70	0.0132
0.80	0.0040
0.90	0.0040
1.00	0.0040
Average precision over all relevant docs	
non-interpolated	0.0637

Document Level Averages	
	Precision
At 5 docs	0.1680
At 10 docs	0.1380
At 15 docs	0.1240
At 20 docs	0.1080
At 30 docs	0.0967
At 100 docs	0.0614
At 200 docs	0.0398
At 500 docs	0.0241
At 1000 docs	0.0143
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1018



Recall-Precision Curve



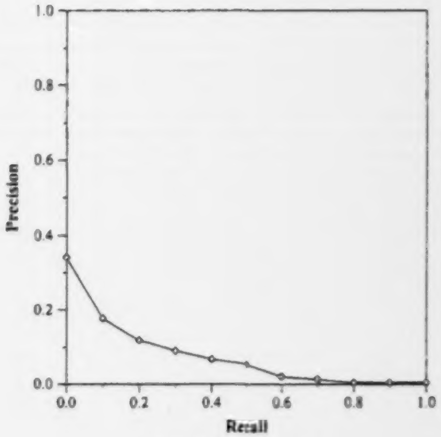
Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	PuShortBase
Run Description	Automatic, content-only, title only
Number of Topics	50

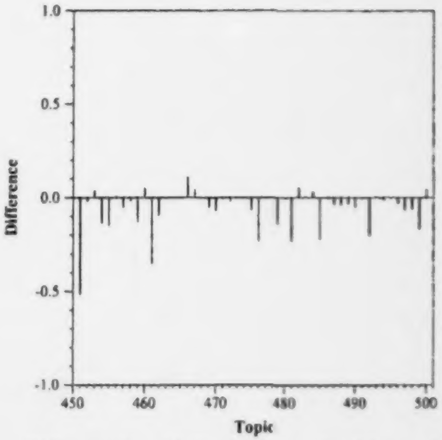
Total number of documents over all topics	
Retrieved:	42538
Relevant:	2617
Rel-ret:	713

Recall Level Precision Averages	
Recall	Precision
0.00	0.3412
0.10	0.1776
0.20	0.1195
0.30	0.0914
0.40	0.0681
0.50	0.0547
0.60	0.0207
0.70	0.0130
0.80	0.0042
0.90	0.0042
1.00	0.0042
Average precision over all relevant docs	
non-interpolated	0.0654

Document Level Averages	
	Precision
At 5 docs	0.1640
At 10 docs	0.1420
At 15 docs	0.1240
At 20 docs	0.1130
At 30 docs	0.0993
At 100 docs	0.0614
At 200 docs	0.0398
At 500 docs	0.0241
At 1000 docs	0.0143
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1031



Recall-Precision Curve



Difference from Median in Average Precision per Topic

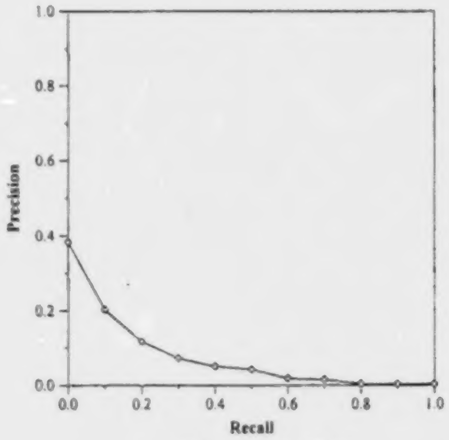
Main web track results — University of Padova

Summary Statistics	
Run Number	PuLongAuth
Run Description	Automatic, content-link, title+desc
Number of Topics	50

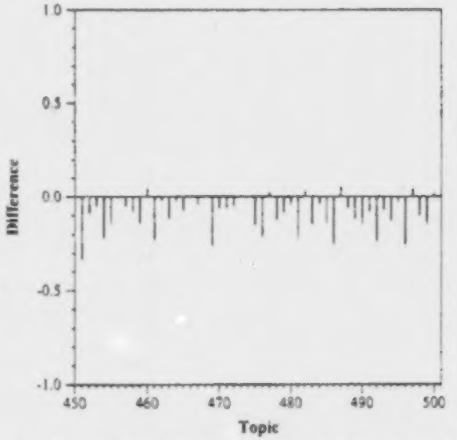
Total number of documents over all topics	
Retrieved:	49282
Relevant:	2617
Rel-ret:	754

Recall Level Precision Averages	
Recall	Precision
0.00	0.3837
0.10	0.2033
0.20	0.1178
0.30	0.0732
0.40	0.0516
0.50	0.0427
0.60	0.0183
0.70	0.0154
0.80	0.0039
0.90	0.0029
1.00	0.0029
Average precision over all relevant docs	
non-interpolated	0.0648

Document Level Averages	
	Precision
At 5 docs	0.2120
At 10 docs	0.1720
At 15 docs	0.1533
At 20 docs	0.1340
At 30 docs	0.1080
At 100 docs	0.0618
At 200 docs	0.0435
At 500 docs	0.0251
At 1000 docs	0.0151
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0995



Recall-Precision Curve



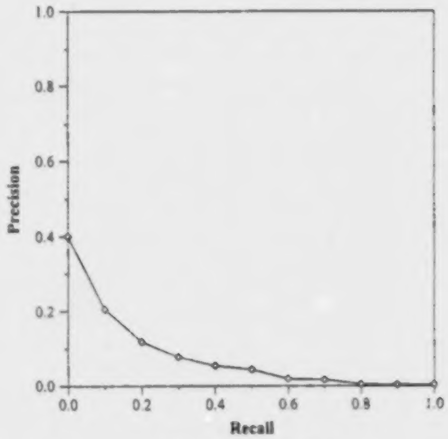
Difference from Median in Average Precision per Topic

Main web track results — University of Padova

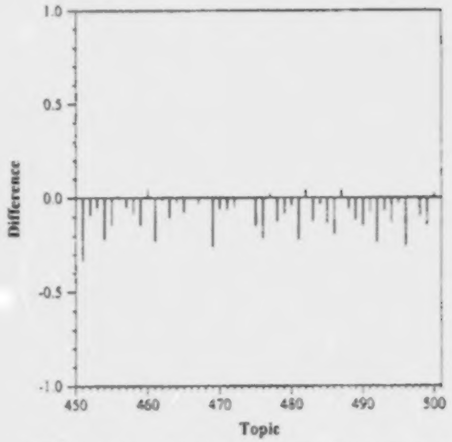
Summary Statistics	
Run Number	PuLongWAuth
Run Description	Automatic, content-link, title+desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49282
Relevant:	2617
Rel-ret:	754

Recall Level Precision Averages	
Recall	Precision
0.00	0.4003
0.10	0.2066
0.20	0.1190
0.30	0.0780
0.40	0.0545
0.50	0.0442
0.60	0.0193
0.70	0.0164
0.80	0.0040
0.90	0.0030
1.00	0.0030
Average precision over all relevant docs	
non-interpolated	0.0660

Document Level Averages	
	Precision
At 5 docs	0.2120
At 10 docs	0.1780
At 15 docs	0.1573
At 20 docs	0.1360
At 30 docs	0.1080
At 100 docs	0.0642
At 200 docs	0.0441
At 500 docs	0.0253
At 1000 docs	0.0151
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1065



Recall-Precision Curve



Difference from Median in Average Precision per Topic

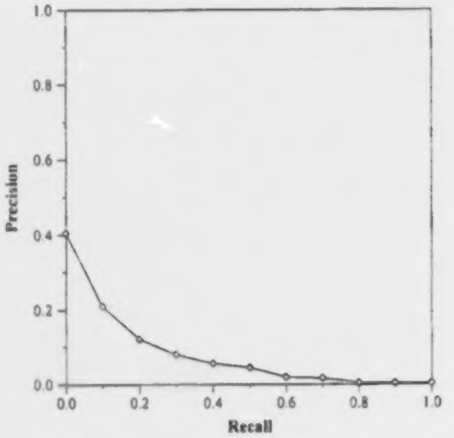
Main web track results — University of Padova

Summary Statistics	
Run Number	PuLongBase
Run Description	Automatic, content-only, title+desc
Number of Topics	50

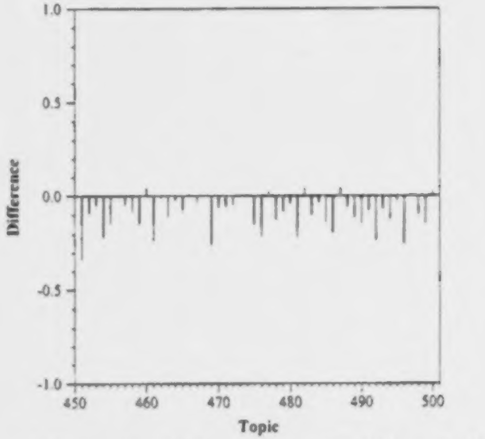
Total number of documents over all topics	
Retrieved:	49282
Relevant:	2617
Rel-ret:	754

Recall Level Precision Averages	
Recall	Precision
0.00	0.4053
0.10	0.2104
0.20	0.1206
0.30	0.0786
0.40	0.0549
0.50	0.0443
0.60	0.0190
0.70	0.0162
0.80	0.0040
0.90	0.0029
1.00	0.0029
Average precision over all relevant docs	
non-interpolated	0.0666

Document Level Averages	
	Precision
At 5 docs	0.2120
At 10 docs	0.1800
At 15 docs	0.1547
At 20 docs	0.1340
At 30 docs	0.1093
At 100 docs	0.0642
At 200 docs	0.0441
At 500 docs	0.0253
At 1000 docs	0.0151
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1071



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	iit00td
Run Description	Automatic, content-only, title+desc
Number of Topics	50

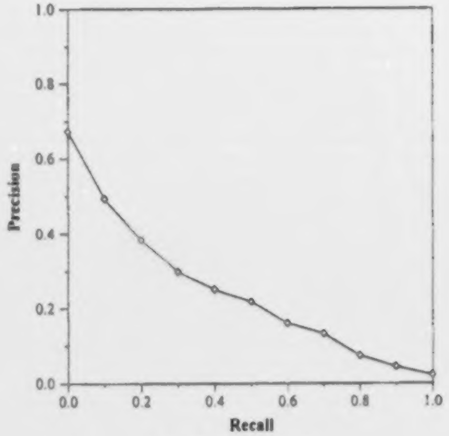
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1878

Recall Level Precision Averages	
Recall	Precision
0.00	0.6733
0.10	0.4946
0.20	0.3836
0.30	0.3002
0.40	0.2522
0.50	0.2187
0.60	0.1598
0.70	0.1318
0.80	0.0731
0.90	0.0453
1.00	0.0232

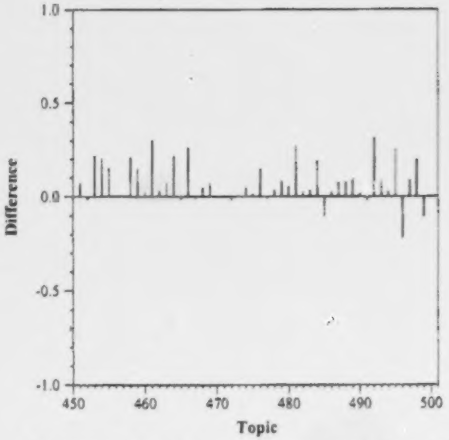
Average precision over all relevant docs	
non-interpolated	0.2293

Document Level Averages	
	Precision
At 5 docs	0.4000
At 10 docs	0.3500
At 15 docs	0.3200
At 20 docs	0.2960
At 30 docs	0.2567
At 100 docs	0.1490
At 200 docs	0.1077
At 500 docs	0.0624
At 1000 docs	0.0376

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2546



Recall-Precision Curve



Difference from Median in Average Precision per Topic

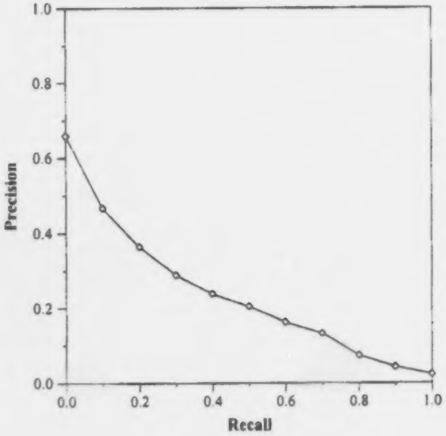
Main web track results — IIT/AAT/NCR

Summary Statistics	
Run Number	iit00tde
Run Description	Automatic, content-only, title+desc
Number of Topics	50

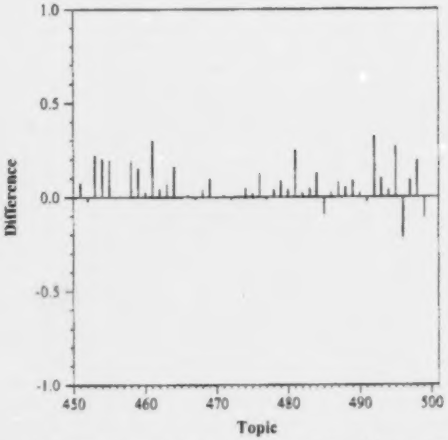
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1869

Recall Level Precision Averages	
Recall	Precision
0.00	0.6597
0.10	0.4675
0.20	0.3644
0.30	0.2896
0.40	0.2382
0.50	0.2051
0.60	0.1627
0.70	0.1323
0.80	0.0728
0.90	0.0436
1.00	0.0241
Average precision over all relevant docs	
non-interpolated	0.2217

Document Level Averages	
	Precision
At 5 docs	0.3880
At 10 docs	0.3460
At 15 docs	0.3160
At 20 docs	0.2970
At 30 docs	0.2587
At 100 docs	0.1522
At 200 docs	0.1071
At 500 docs	0.0621
At 1000 docs	0.0374
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2505



Recall-Precision Curve



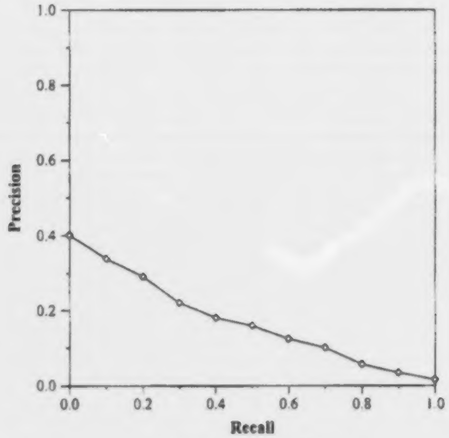
Difference from Median in Average Precision per Topic

Main web track results — IIT/AAT/NCR

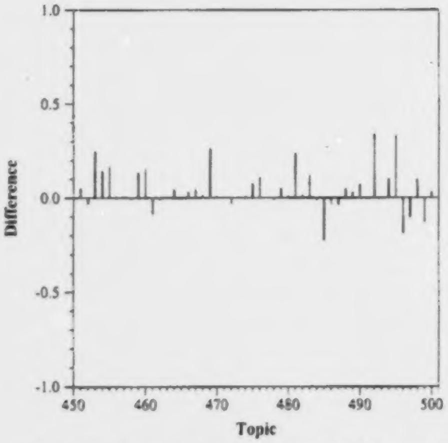
Summary Statistics	
Run Number	iit00t
Run Description	Automatic, content-only, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1511

Recall Level Precision Averages	
Recall	Precision
0.00	0.4015
0.10	0.3388
0.20	0.2919
0.30	0.2215
0.40	0.1808
0.50	0.1589
0.60	0.1239
0.70	0.1010
0.80	0.0565
0.90	0.0344
1.00	0.0159
Average precision over all relevant docs	
non-interpolated	0.1627

Document Level Averages	
	Precision
At 5 docs	0.2720
At 10 docs	0.2500
At 15 docs	0.2360
At 20 docs	0.2210
At 30 docs	0.1973
At 100 docs	0.1244
At 200 docs	0.0882
At 500 docs	0.0494
At 1000 docs	0.0302
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1868



Recall-Precision Curve



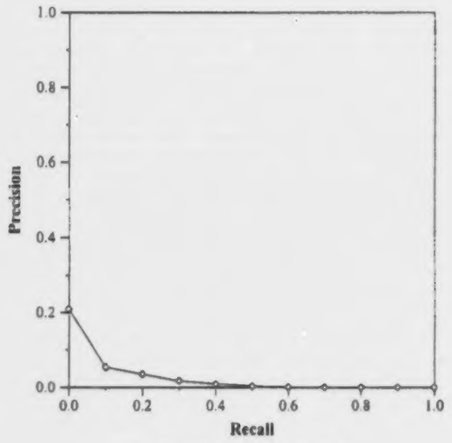
Difference from Median in Average Precision per Topic

Main web track results — CWI, The Netherlands

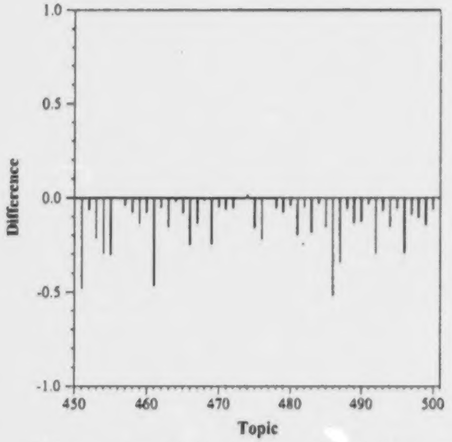
Summary Statistics	
Run Number	CWI0001
Run Description	Automatic, content-only, title+desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49578
Relevant:	2617
Rel-ret:	630

Recall Level Precision Averages	
Recall	Precision
0.00	0.2099
0.10	0.0545
0.20	0.0365
0.30	0.0186
0.40	0.0096
0.50	0.0034
0.60	0.0009
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0174

Document Level Averages	
	Precision
At 5 docs	0.0600
At 10 docs	0.0540
At 15 docs	0.0467
At 20 docs	0.0450
At 30 docs	0.0487
At 100 docs	0.0444
At 200 docs	0.0316
At 500 docs	0.0189
At 1000 docs	0.0126
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0367



Recall-Precision Curve



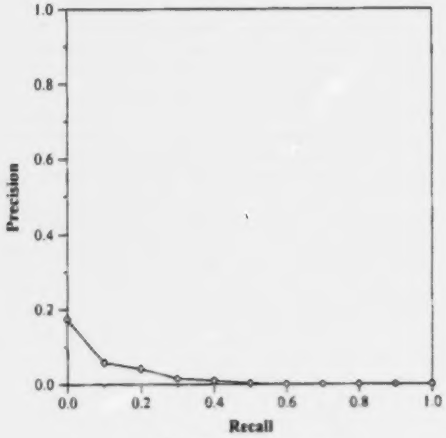
Difference from Median in Average Precision per Topic

Main web track results — CWI, The Netherlands

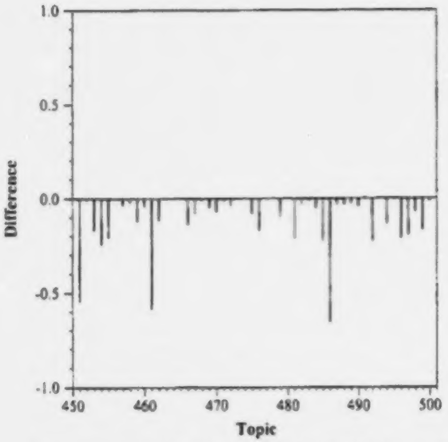
Summary Statistics	
Run Number	CWI0000
Run Description	Automatic, content-only, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	44776
Relevant:	2617
Rel-ret:	588

Recall Level Precision Averages	
Recall	Precision
0.00	0.1742
0.10	0.0581
0.20	0.0420
0.30	0.0157
0.40	0.0099
0.50	0.0025
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0176

Document Level Averages	
	Precision
At 5 docs	0.0760
At 10 docs	0.0660
At 15 docs	0.0587
At 20 docs	0.0570
At 30 docs	0.0547
At 100 docs	0.0372
At 200 docs	0.0315
At 500 docs	0.0177
At 1000 docs	0.0118
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0413



Recall-Precision Curve



Difference from Median in Average Precision per Topic

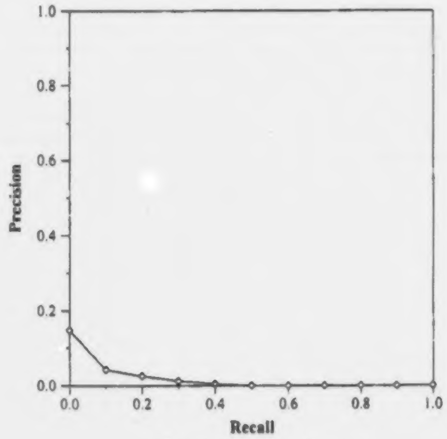
Main web track results — CWI, The Netherlands

Summary Statistics	
Run Number	CWI0002
Run Description	Automatic, content-only, title+desc+narr
Number of Topics	50

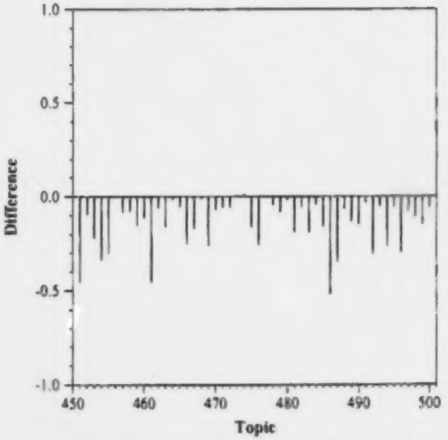
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	448

Recall Level Precision Averages	
Recall	Precision
0.00	0.1471
0.10	0.0416
0.20	0.0254
0.30	0.0125
0.40	0.0044
0.50	0.0006
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0122

Document Level Averages	
	Precision
At 5 docs	0.0280
At 10 docs	0.0380
At 15 docs	0.0320
At 20 docs	0.0370
At 30 docs	0.0347
At 100 docs	0.0278
At 200 docs	0.0217
At 500 docs	0.0136
At 1000 docs	0.0090
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0327



Recall-Precision Curve



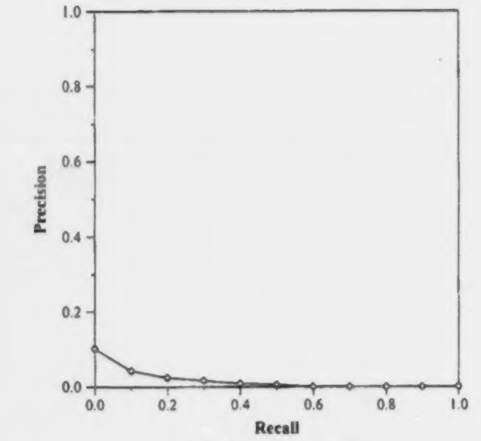
Difference from Median in Average Precision per Topic

Main web track results — CWI, The Netherlands

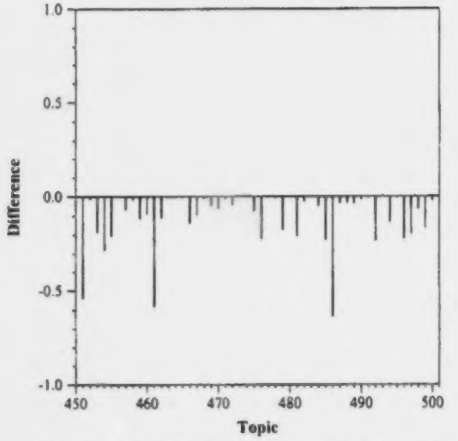
Summary Statistics	
Run Number	CWI0010
Run Description	Automatic, content-only, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	44776
Relevant:	2617
Rel-ret:	623

Recall Level Precision Averages	
Recall	Precision
0.00	0.1004
0.10	0.0420
0.20	0.0247
0.30	0.0168
0.40	0.0089
0.50	0.0056
0.60	0.0002
0.70	0.0002
0.80	0.0000
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0125

Document Level Averages	
	Precision
At 5 docs	0.0240
At 10 docs	0.0240
At 15 docs	0.0320
At 20 docs	0.0360
At 30 docs	0.0393
At 100 docs	0.0276
At 200 docs	0.0242
At 500 docs	0.0163
At 1000 docs	0.0125
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0299



Recall-Precision Curve



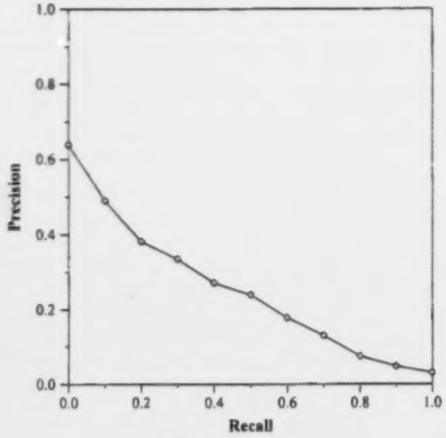
Difference from Median in Average Precision per Topic

Main web track results — Hummingbird Communications Ltd.

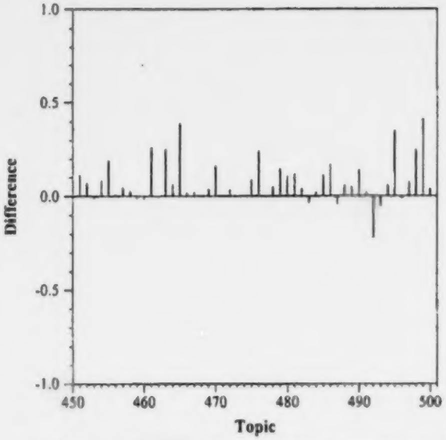
Summary Statistics	
Run Number	hum9tdn
Run Description	Automatic, content-only, title+desc+narr
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1882

Recall Level Precision Averages	
Recall	Precision
0.00	0.6391
0.10	0.4909
0.20	0.3817
0.30	0.3341
0.40	0.2715
0.50	0.2401
0.60	0.1783
0.70	0.1297
0.80	0.0737
0.90	0.0468
1.00	0.0287
Average precision over all relevant docs	
non-interpolated	0.2335

Document Level Averages	
	Precision
At 5 docs	0.4200
At 10 docs	0.3520
At 15 docs	0.3053
At 20 docs	0.2740
At 30 docs	0.2367
At 100 docs	0.1588
At 200 docs	0.1134
At 500 docs	0.0641
At 1000 docs	0.0376
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2707



Recall-Precision Curve



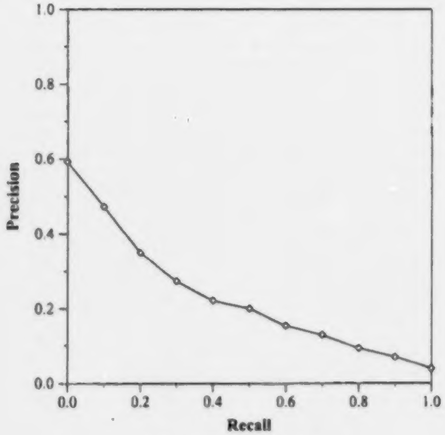
Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	pir0Wttd
Run Description	Automatic, content-only, title+desc
Number of Topics	50

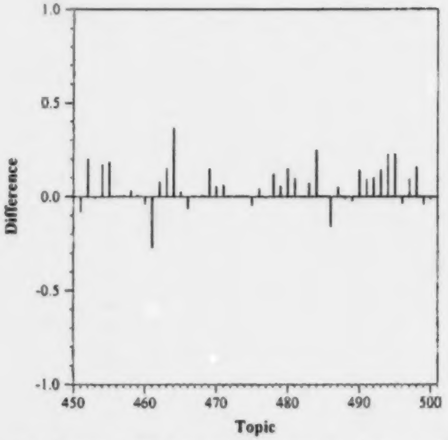
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	2005

Recall Level Precision Averages	
Recall	Precision
0.00	0.5928
0.10	0.4730
0.20	0.3493
0.30	0.2728
0.40	0.2222
0.50	0.2008
0.60	0.1550
0.70	0.1295
0.80	0.0939
0.90	0.0697
1.00	0.0391
Average precision over all relevant docs	
non-interpolated	0.2097

Document Level Averages	
	Precision
At 5 docs	0.3240
At 10 docs	0.3180
At 15 docs	0.2787
At 20 docs	0.2640
At 30 docs	0.2327
At 100 docs	0.1558
At 200 docs	0.1202
At 500 docs	0.0690
At 1000 docs	0.0401
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2125



Recall-Precision Curve



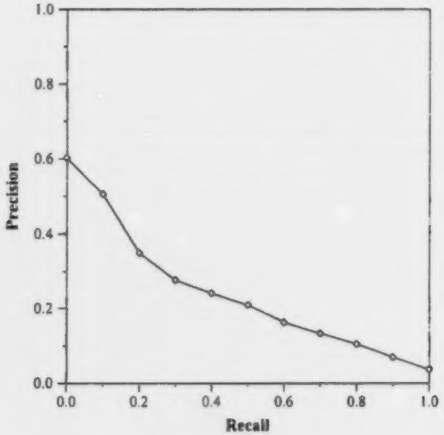
Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	pir0Watd
Run Description	Automatic, content-only, title+desc+narr
Number of Topics	50

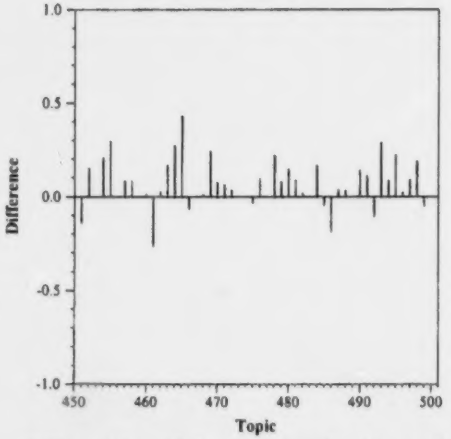
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	2011

Recall Level Precision Averages	
Recall	Precision
0.00	0.6029
0.10	0.5061
0.20	0.3493
0.30	0.2781
0.40	0.2419
0.50	0.2106
0.60	0.1638
0.70	0.1337
0.80	0.1051
0.90	0.0703
1.00	0.0368
Average precision over all relevant docs	
non-interpolated	0.2209

Document Level Averages	
	Precision
At 5 docs	0.3360
At 10 docs	0.2980
At 15 docs	0.2827
At 20 docs	0.2750
At 30 docs	0.2433
At 100 docs	0.1648
At 200 docs	0.1261
At 500 docs	0.0698
At 1000 docs	0.0402
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2275



Recall-Precision Curve

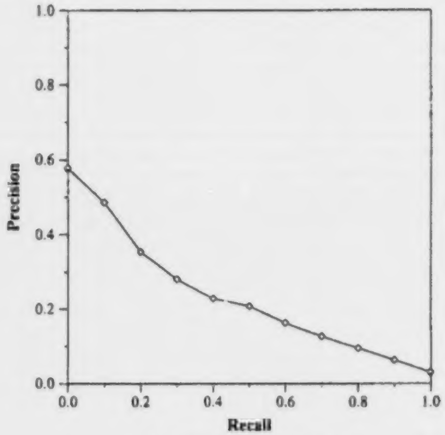


Difference from Median in Average Precision per Topic

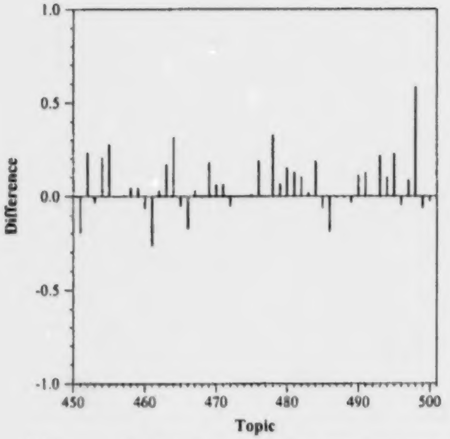
Summary Statistics	
Run Number	pir0Wtd2
Run Description	Automatic, content-only, title+desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	2010

Recall Level Precision Averages	
Recall	Precision
0.00	0.5778
0.10	0.4861
0.20	0.3538
0.30	0.2811
0.40	0.2285
0.50	0.2077
0.60	0.1627
0.70	0.1259
0.80	0.0944
0.90	0.0629
1.00	0.0306
Average precision over all relevant docs	
non-interpolated	0.2164

Document Level Averages	
	Precision
At 5 docs	0.3400
At 10 docs	0.3020
At 15 docs	0.2747
At 20 docs	0.2570
At 30 docs	0.2393
At 100 docs	0.1610
At 200 docs	0.1215
At 500 docs	0.0671
At 1000 docs	0.0402
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2242



Recall-Precision Curve



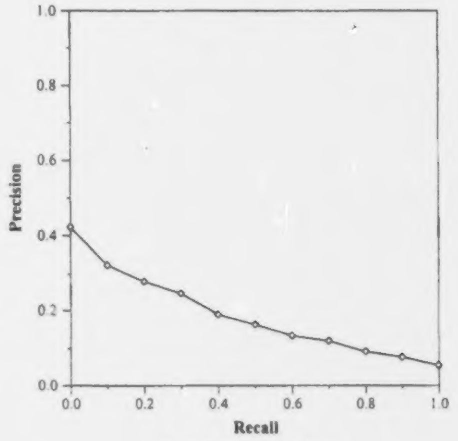
Difference from Median in Average Precision per Topic

Main web track results — Queens College, CUNY

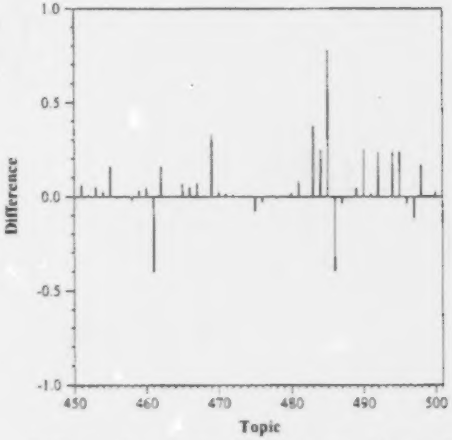
Summary Statistics	
Run Number	pir0Wt1
Run Description	Automatic, content-only, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1518

Recall Level Precision Averages	
Recall	Precision
0.00	0.4231
0.10	0.3228
0.20	0.2792
0.30	0.2466
0.40	0.1893
0.50	0.1620
0.60	0.1327
0.70	0.1193
0.80	0.0917
0.90	0.0768
1.00	0.0553
Average precision over all relevant docs	
non-interpolated	0.1750

Document Level Averages	
	Precision
At 5 docs	0.2200
At 10 docs	0.2180
At 15 docs	0.2120
At 20 docs	0.1920
At 30 docs	0.1773
At 100 docs	0.1240
At 200 docs	0.0929
At 500 docs	0.0520
At 1000 docs	0.0304
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1893



Recall-Precision Curve



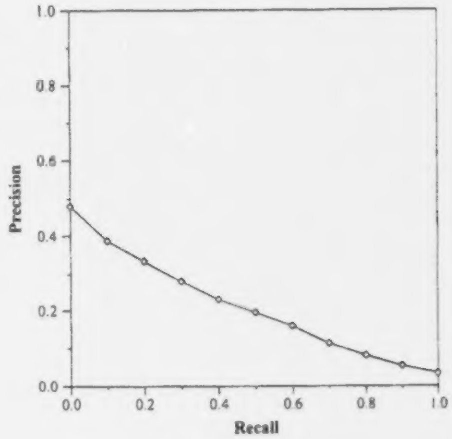
Difference from Median in Average Precision per Topic

Main web track results — Hummingbird Communications Ltd.

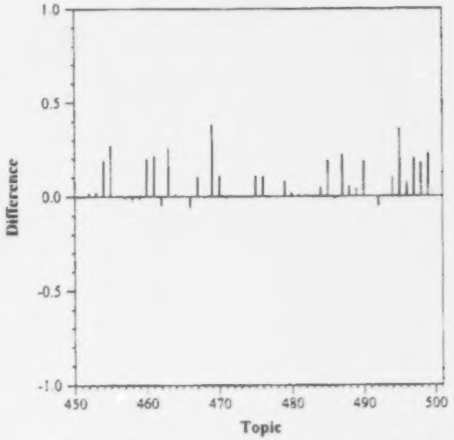
Summary Statistics	
Run Number	hum9te
Run Description	Automatic, content-only, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1682

Recall Level Precision Averages	
Recall	Precision
0.00	0.4802
0.10	0.3872
0.20	0.3331
0.30	0.2808
0.40	0.2315
0.50	0.1951
0.60	0.1582
0.70	0.1124
0.80	0.0805
0.90	0.0532
1.00	0.0343
Average precision over all relevant docs	
non-interpolated	0.1970

Document Level Averages	
	Precision
At 5 docs	0.3240
At 10 docs	0.2540
At 15 docs	0.2400
At 20 docs	0.2150
At 30 docs	0.1933
At 100 docs	0.1334
At 200 docs	0.0979
At 500 docs	0.0561
At 1000 docs	0.0336
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2113



Recall-Precision Curve



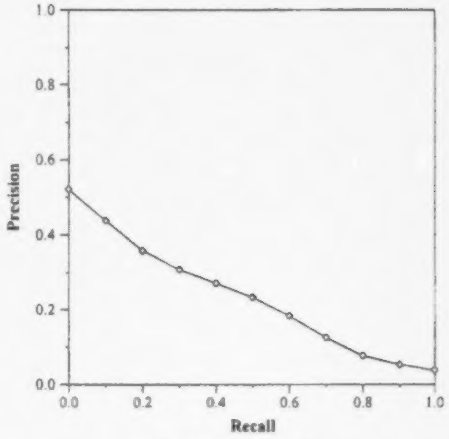
Difference from Median in Average Precision per Topic

Main web track results — Hummingbird Communications Ltd.

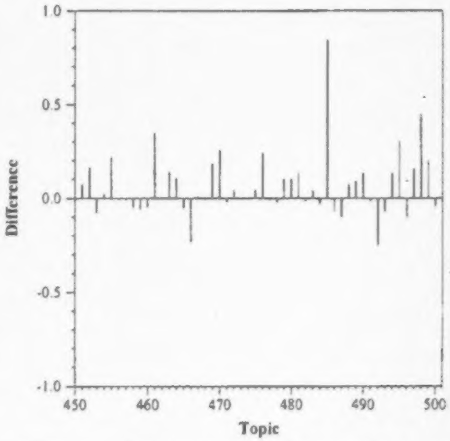
Summary Statistics	
Run Number	hum9tde
Run Description	Automatic, content-only, title+desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1842

Recall Level Precision Averages	
Recall	Precision
0.00	0.5217
0.10	0.4383
0.20	0.3579
0.30	0.3082
0.40	0.2712
0.50	0.2334
0.60	0.1828
0.70	0.1240
0.80	0.0770
0.90	0.0537
1.00	0.0392
Average precision over all relevant docs	
non-interpolated	0.2217

Document Level Averages	
	Precision
At 5 docs	0.3720
At 10 docs	0.2940
At 15 docs	0.2627
At 20 docs	0.2400
At 30 docs	0.2153
At 100 docs	0.1424
At 200 docs	0.1032
At 500 docs	0.0609
At 1000 docs	0.0368
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2440



Recall-Precision Curve



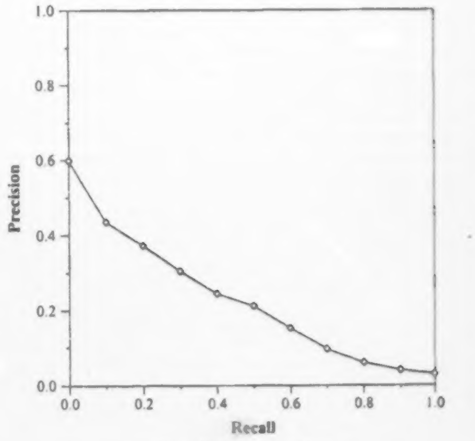
Difference from Median in Average Precision per Topic

Main web track results — Hummingbird Communications Ltd.

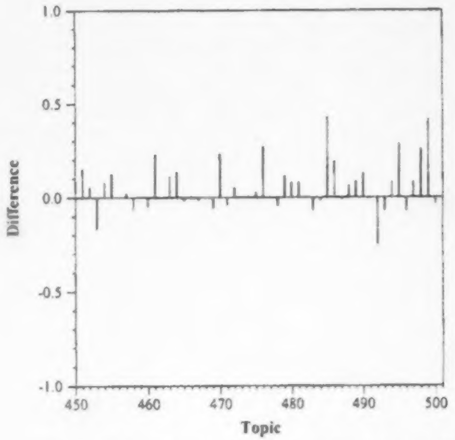
Summary Statistics	
Run Number	hum9td4
Run Description	Automatic, content-only, title+desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1699

Recall Level Precision Averages	
Recall	Precision
0.00	0.5990
0.10	0.4359
0.20	0.3722
0.30	0.3051
0.40	0.2450
0.50	0.2120
0.60	0.1520
0.70	0.0949
0.80	0.0591
0.90	0.0398
1.00	0.0296
Average precision over all relevant docs	
non-interpolated	0.2115

Document Level Averages	
	Precision
At 5 docs	0.3760
At 10 docs	0.3080
At 15 docs	0.2733
At 20 docs	0.2480
At 30 docs	0.2133
At 100 docs	0.1444
At 200 docs	0.0986
At 500 docs	0.0560
At 1000 docs	0.0340
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2558



Recall-Precision Curve

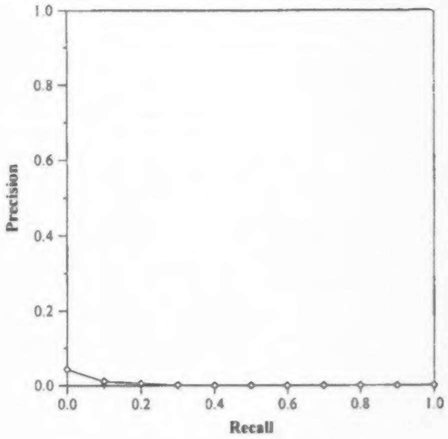


Difference from Median in Average Precision per Topic

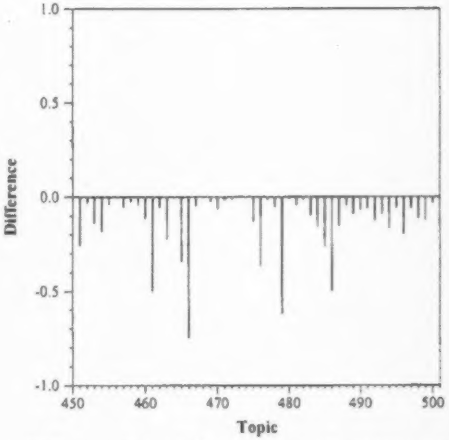
Summary Statistics	
Run Number	rmitWFGweb
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	47074
Relevant:	2617
Rel-ret:	170

Recall Level Precision Averages	
Recall	Precision
0.00	0.0436
0.10	0.0114
0.20	0.0056
0.30	0.0015
0.40	0.0005
0.50	0.0005
0.60	0.0002
0.70	0.0002
0.80	0.0001
0.90	0.0001
1.00	0.0001
Average precision over all relevant docs	
non-interpolated	0.0040

Document Level Averages	
	Precision
At 5 docs	0.0040
At 10 docs	0.0100
At 15 docs	0.0120
At 20 docs	0.0110
At 30 docs	0.0120
At 100 docs	0.0098
At 200 docs	0.0073
At 500 docs	0.0052
At 1000 docs	0.0034
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0089



Recall-Precision Curve

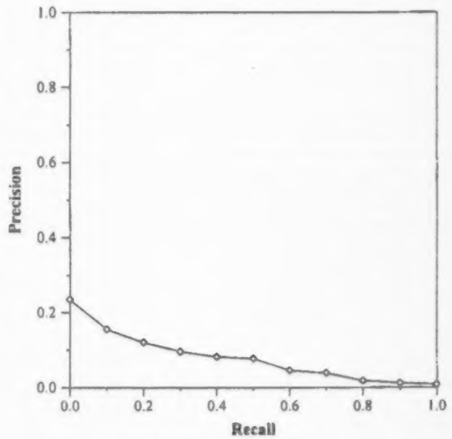


Difference from Median in Average Precision per Topic

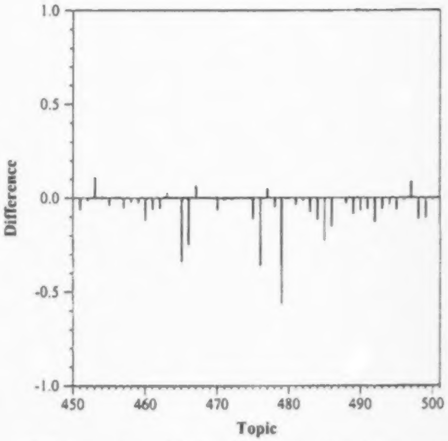
Summary Statistics	
Run Number	rmitNFGweb
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	47091
Relevant:	2617
Rel-ret:	858

Recall Level Precision Averages	
Recall	Precision
0.00	0.2355
0.10	0.1549
0.20	0.1201
0.30	0.0960
0.40	0.0822
0.50	0.0772
0.60	0.0456
0.70	0.0385
0.80	0.0189
0.90	0.0124
1.00	0.0083
Average precision over all relevant docs	
non-interpolated	0.0707

Document Level Averages	
	Precision
At 5 docs	0.0840
At 10 docs	0.0880
At 15 docs	0.0773
At 20 docs	0.0700
At 30 docs	0.0660
At 100 docs	0.0532
At 200 docs	0.0394
At 500 docs	0.0249
At 1000 docs	0.0172
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0884



Recall-Precision Curve

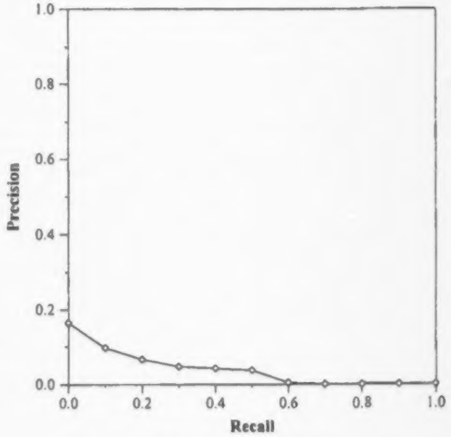


Difference from Median in Average Precision per Topic

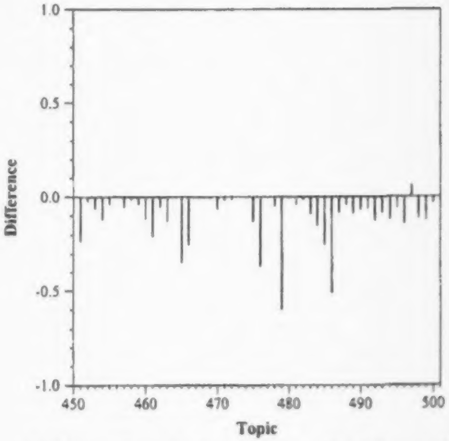
Summary Statistics	
Run Number	rmitWFLweb
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	47074
Relevant:	2617
Rel-ret:	539

Recall Level Precision Averages	
Recall	Precision
0.00	0.1648
0.10	0.0970
0.20	0.0667
0.30	0.0475
0.40	0.0423
0.50	0.0368
0.60	0.0047
0.70	0.0012
0.80	0.0012
0.90	0.0009
1.00	0.0007
Average precision over all relevant docs	
non-interpolated	0.0341

Document Level Averages	
	Precision
At 5 docs	0.0440
At 10 docs	0.0440
At 15 docs	0.0387
At 20 docs	0.0340
At 30 docs	0.0353
At 100 docs	0.0278
At 200 docs	0.0228
At 500 docs	0.0153
At 1000 docs	0.0108
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0439



Recall-Precision Curve



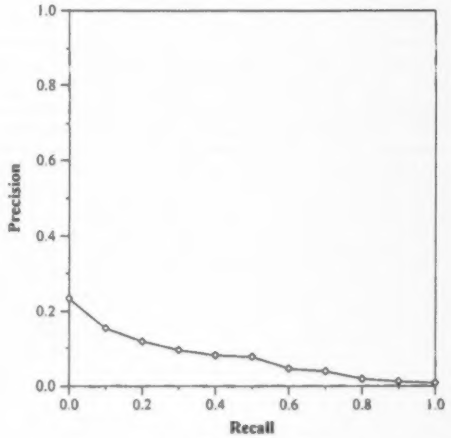
Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	rmitNFLweb
Run Description	Manual
Number of Topics	50

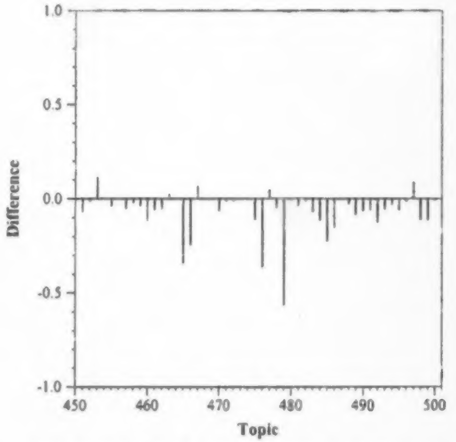
Total number of documents over all topics	
Retrieved:	47091
Relevant:	2617
Rel-ret:	854

Recall Level Precision Averages	
Recall	Precision
0.00	0.2354
0.10	0.1535
0.20	0.1183
0.30	0.0947
0.40	0.0809
0.50	0.0770
0.60	0.0454
0.70	0.0385
0.80	0.0187
0.90	0.0123
1.00	0.0082
Average precision over all relevant docs	
non-interpolated	0.0702

Document Level Averages	
	Precision
At 5 docs	0.0920
At 10 docs	0.0900
At 15 docs	0.0800
At 20 docs	0.0700
At 30 docs	0.0667
At 100 docs	0.0516
At 200 docs	0.0391
At 500 docs	0.0248
At 1000 docs	0.0171
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0850



Recall-Precision Curve



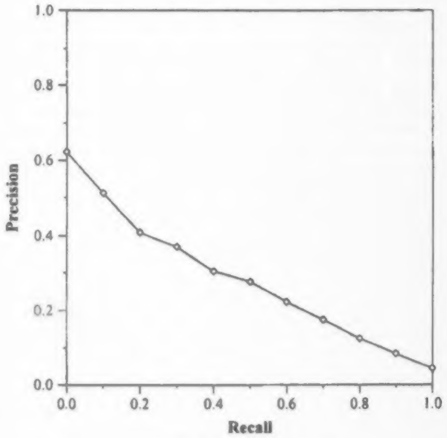
Difference from Median in Average Precision per Topic

Main web track results — Justsystem Corporation

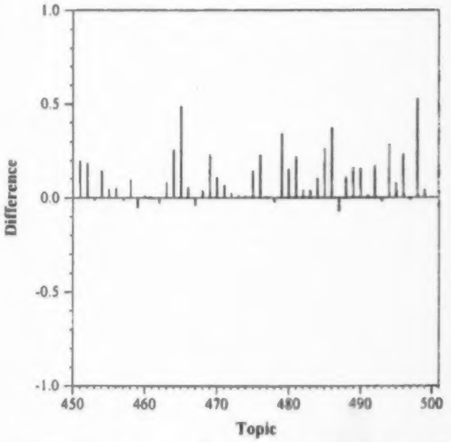
Summary Statistics	
Run Number	jscbt9wll1
Run Description	Automatic, content-link, title+desc+narr
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1916

Recall Level Precision Averages	
Recall	Precision
0.00	0.6230
0.10	0.5123
0.20	0.4089
0.30	0.3693
0.40	0.3035
0.50	0.2763
0.60	0.2232
0.70	0.1753
0.80	0.1249
0.90	0.0832
1.00	0.0430
Average precision over all relevant docs	
non-interpolated	0.2659

Document Level Averages	
	Precision
At 5 docs	0.3880
At 10 docs	0.3440
At 15 docs	0.3067
At 20 docs	0.2920
At 30 docs	0.2580
At 100 docs	0.1680
At 200 docs	0.1198
At 500 docs	0.0663
At 1000 docs	0.0383
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2812



Recall-Precision Curve



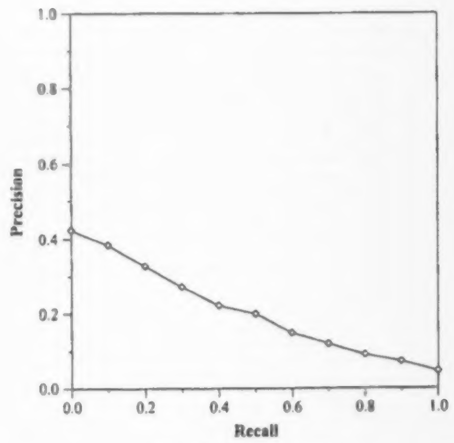
Difference from Median in Average Precision per Topic

Main web track results — Justsystem Corporation

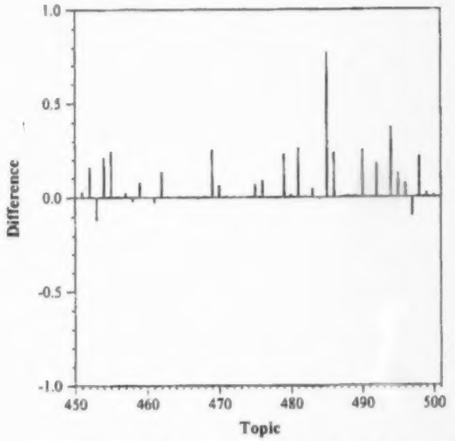
Summary Statistics	
Run Number	jscbt9wls1
Run Description	Automatic, content-link, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1526

Recall Level Precision Averages	
Recall	Precision
0.00	0.4243
0.10	0.3841
0.20	0.3276
0.30	0.2717
0.40	0.2220
0.50	0.2000
0.60	0.1501
0.70	0.1197
0.80	0.0916
0.90	0.0723
1.00	0.0461
Average precision over all relevant docs	
non-interpolated	0.2000

Document Level Averages	
	Precision
At 5 docs	0.2720
At 10 docs	0.2520
At 15 docs	0.2267
At 20 docs	0.2110
At 30 docs	0.1893
At 100 docs	0.1356
At 200 docs	0.0955
At 500 docs	0.0515
At 1000 docs	0.0305
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2219



Recall-Precision Curve



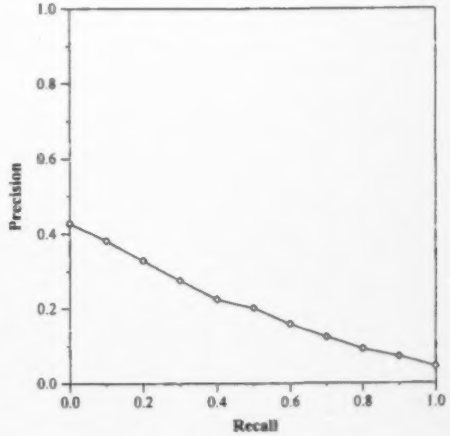
Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	jscbt9wcs1
Run Description	Automatic, content-only, title only
Number of Topics	50

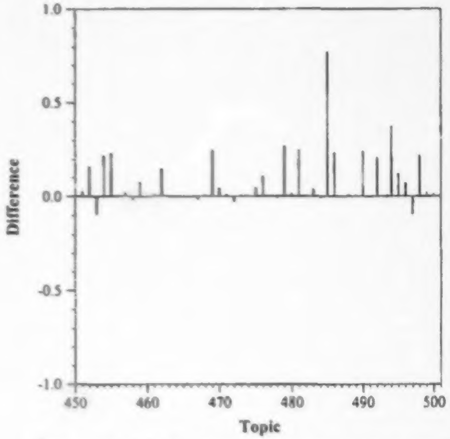
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1517

Recall Level Precision Averages	
Recall	Precision
0.00	0.4285
0.10	0.3820
0.20	0.3291
0.30	0.2755
0.40	0.2250
0.50	0.2019
0.60	0.1598
0.70	0.1247
0.80	0.0930
0.90	0.0730
1.00	0.0459
Average precision over all relevant docs	
non-interpolated	0.2011

Document Level Averages	
	Precision
At 5 docs	0.2760
At 10 docs	0.2380
At 15 docs	0.2213
At 20 docs	0.2040
At 30 docs	0.1867
At 100 docs	0.1378
At 200 docs	0.0962
At 500 docs	0.0512
At 1000 docs	0.0303
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2175



Recall-Precision Curve



Difference from Median in Average Precision per Topic

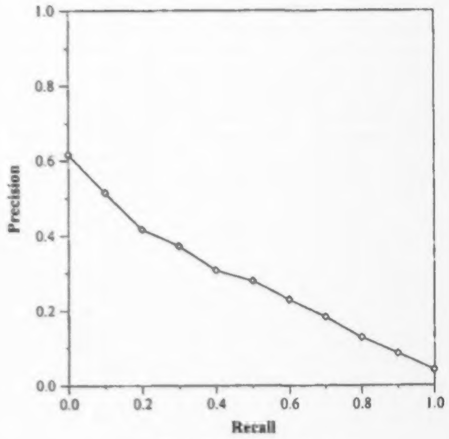
Main web track results — Justsystem Corporation

Summary Statistics	
Run Number	jscbt9wcl1
Run Description	Automatic, content-only, title+desc+narr
Number of Topics	50

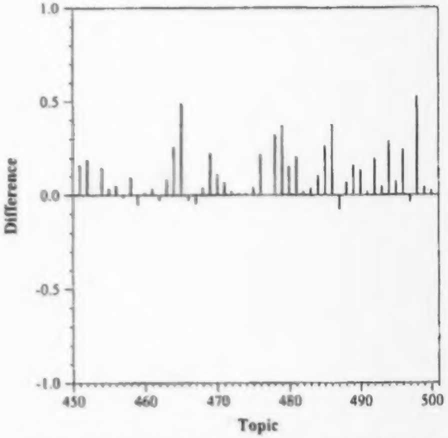
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1941

Recall Level Precision Averages	
Recall	Precision
0.00	0.6164
0.10	0.5148
0.20	0.4179
0.30	0.3722
0.40	0.3079
0.50	0.2793
0.60	0.2299
0.70	0.1847
0.80	0.1278
0.90	0.0852
1.00	0.0413
Average precision over all relevant docs	
non-interpolated	0.2687

Document Level Averages	
	Precision
At 5 docs	0.3880
At 10 docs	0.3420
At 15 docs	0.3093
At 20 docs	0.3030
At 30 docs	0.2700
At 100 docs	0.1782
At 200 docs	0.1244
At 500 docs	0.0674
At 1000 docs	0.0388
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2841



Recall-Precision Curve

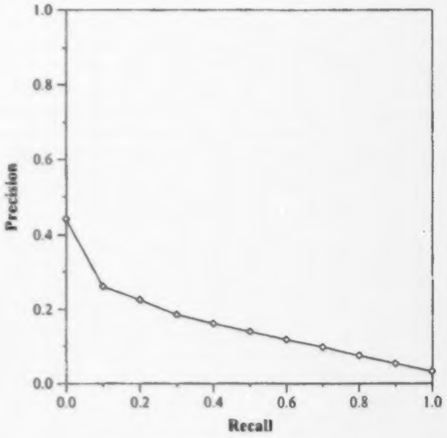


Difference from Median in Average Precision per Topic

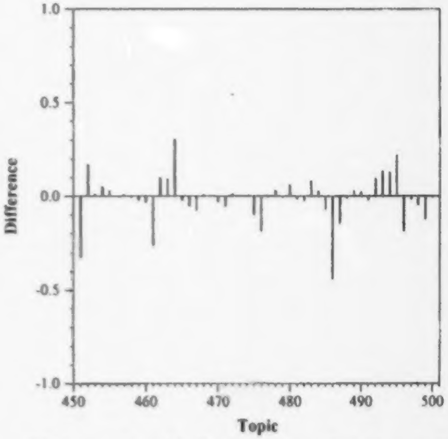
Summary Statistics	
Run Number	pir0WTTD
Run Description	Automatic, content-link, title+desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	2005

Recall Level Precision Averages	
Recall	Precision
0.00	0.4425
0.10	0.2613
0.20	0.2261
0.30	0.1863
0.40	0.1628
0.50	0.1410
0.60	0.1188
0.70	0.0974
0.80	0.0754
0.90	0.0536
1.00	0.0323
Average precision over all relevant docs	
non-interpolated	0.1418

Document Level Averages	
	Precision
At 5 docs	0.2000
At 10 docs	0.1800
At 15 docs	0.1840
At 20 docs	0.1740
At 30 docs	0.1680
At 100 docs	0.1362
At 200 docs	0.1032
At 500 docs	0.0657
At 1000 docs	0.0401
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1439



Recall-Precision Curve



Difference from Median in Average Precision per Topic

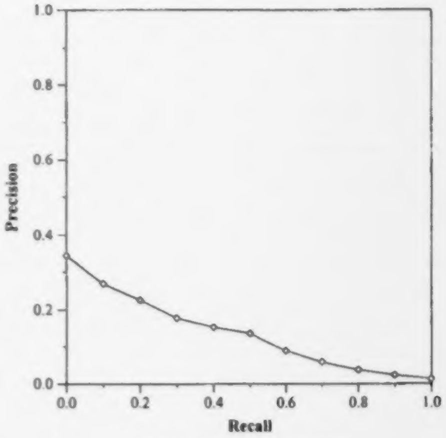
Main web track results — Sabir Research, Inc.

Summary Statistics	
Run Number	Sab9web1
Run Description	Automatic, content-only, title only
Number of Topics	50

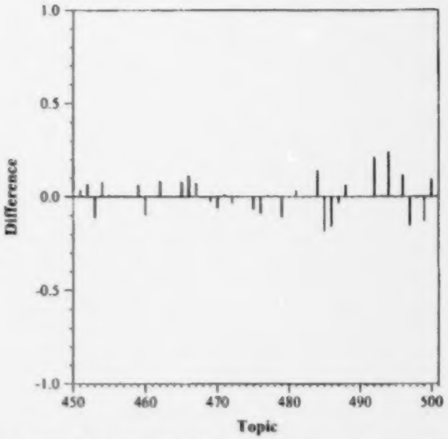
Total number of documents over all topics	
Retrieved:	48002
Relevant:	2617
Rel-ret:	1250

Recall Level Precision Averages	
Recall	Precision
0.00	0.3443
0.10	0.2682
0.20	0.2257
0.30	0.1783
0.40	0.1531
0.50	0.1361
0.60	0.0888
0.70	0.0585
0.80	0.0374
0.90	0.0231
1.00	0.0125
Average precision over all relevant docs	
non-interpolated	0.1265

Document Level Averages	
	Precision
At 5 docs	0.2080
At 10 docs	0.1820
At 15 docs	0.1667
At 20 docs	0.1550
At 30 docs	0.1487
At 100 docs	0.0964
At 200 docs	0.0675
At 500 docs	0.0399
At 1000 docs	0.0250
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1518



Recall-Precision Curve



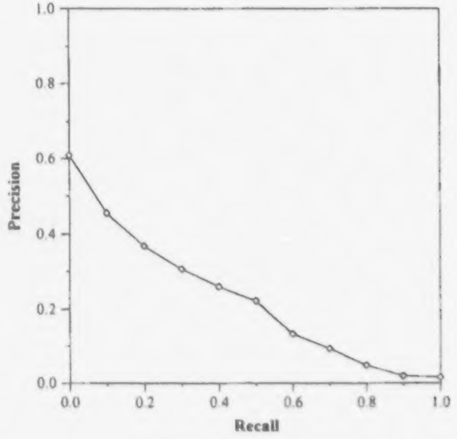
Difference from Median in Average Precision per Topic

Main web track results — Sabir Research, Inc.

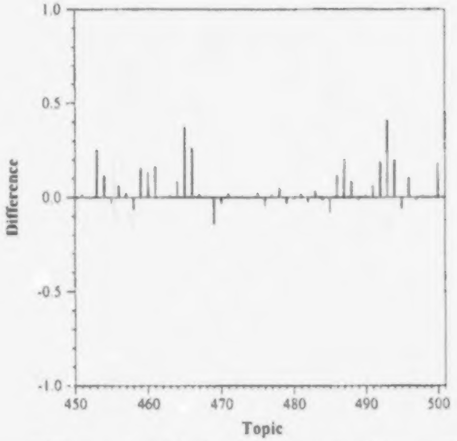
Summary Statistics	
Run Number	Sab9web2
Run Description	Automatic, content-only, title+desc+narr
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1468

Recall Level Precision Averages	
Recall	Precision
0.00	0.6094
0.10	0.4557
0.20	0.3678
0.30	0.3055
0.40	0.2586
0.50	0.2220
0.60	0.1338
0.70	0.0928
0.80	0.0479
0.90	0.0197
1.00	0.0156
Average precision over all relevant docs	
non-interpolated	0.2122

Document Level Averages	
	Precision
At 5 docs	0.3760
At 10 docs	0.3400
At 15 docs	0.3173
At 20 docs	0.2920
At 30 docs	0.2553
At 100 docs	0.1466
At 200 docs	0.0952
At 500 docs	0.0504
At 1000 docs	0.0294
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2463



Recall-Precision Curve



Difference from Median in Average Precision per Topic

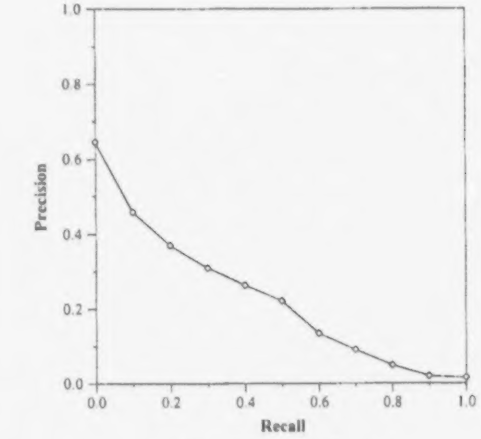
Main web track results — Sabir Research, Inc.

Summary Statistics	
Run Number	Sab9web3
Run Description	Automatic, content-only, title+desc+narr
Number of Topics	50

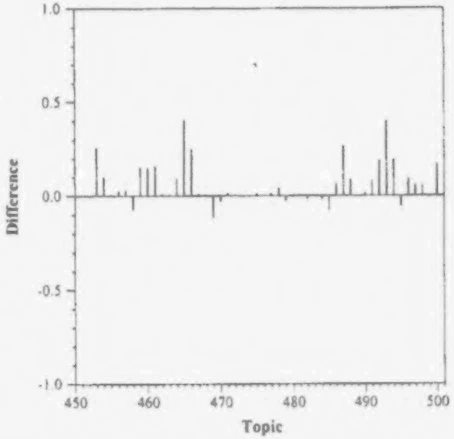
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1456

Recall Level Precision Averages	
Recall	Precision
0.00	0.6454
0.10	0.4582
0.20	0.3704
0.30	0.3097
0.40	0.2634
0.50	0.2214
0.60	0.1350
0.70	0.0909
0.80	0.0488
0.90	0.0190
1.00	0.0145
Average precision over all relevant docs	
non-interpolated	0.2159

Document Level Averages	
	Precision
At 5 docs	0.4000
At 10 docs	0.3460
At 15 docs	0.3253
At 20 docs	0.2970
At 30 docs	0.2493
At 100 docs	0.1450
At 200 docs	0.0955
At 500 docs	0.0504
At 1000 docs	0.0291
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2564



Recall-Precision Curve



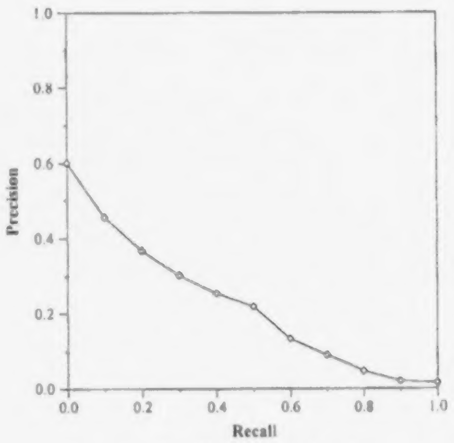
Difference from Median in Average Precision per Topic

Main web track results — Sabir Research, Inc.

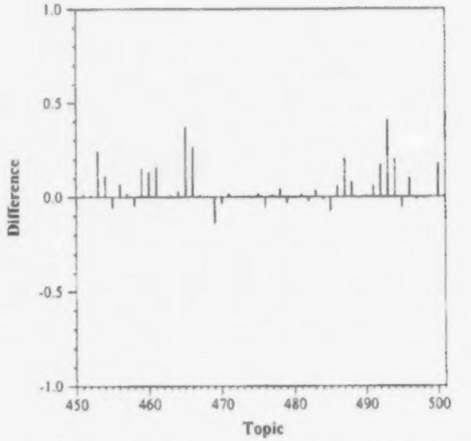
Summary Statistics	
Run Number	Sab9web4
Run Description	Automatic, content-only, title+desc+narr
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1476

Recall Level Precision Averages	
Recall	Precision
0.00	0.6012
0.10	0.4569
0.20	0.3681
0.30	0.3033
0.40	0.2541
0.50	0.2191
0.60	0.1333
0.70	0.0906
0.80	0.0473
0.90	0.0204
1.00	0.0157
Average precision over all relevant docs	
non-interpolated	0.2091

Document Level Averages	
	Precision
At 5 docs	0.3760
At 10 docs	0.3420
At 15 docs	0.3147
At 20 docs	0.2940
At 30 docs	0.2533
At 100 docs	0.1470
At 200 docs	0.0952
At 500 docs	0.0504
At 1000 docs	0.0295
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2485



Recall-Precision Curve



Difference from Median in Average Precision per Topic

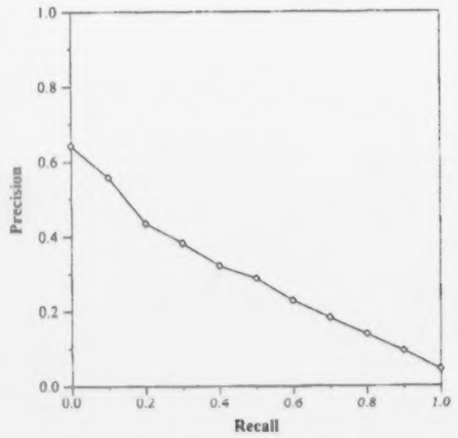
Main web track results — Justsystem Corporation

Summary Statistics	
Run Number	jscbt9wll2
Run Description	Automatic, content-link, title+desc+narr
Number of Topics	50

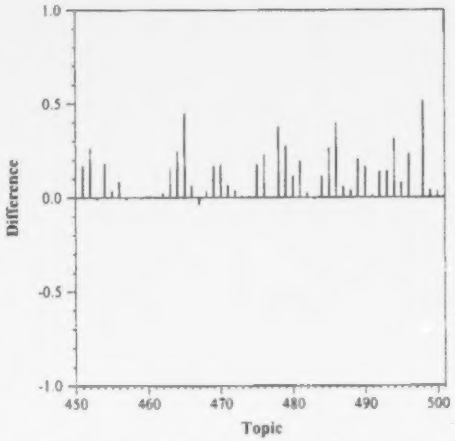
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1971

Recall Level Precision Averages	
Recall	Precision
0.00	0.6426
0.10	0.5576
0.20	0.4353
0.30	0.3836
0.40	0.3224
0.50	0.2879
0.60	0.2278
0.70	0.1827
0.80	0.1388
0.90	0.0945
1.00	0.0447
Average precision over all relevant docs	
non-interpolated	0.2801

Document Level Averages	
	Precision
At 5 docs	0.4000
At 10 docs	0.3580
At 15 docs	0.3347
At 20 docs	0.3060
At 30 docs	0.2880
At 100 docs	0.1812
At 200 docs	0.1291
At 500 docs	0.0695
At 1000 docs	0.0394
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3054



Recall-Precision Curve



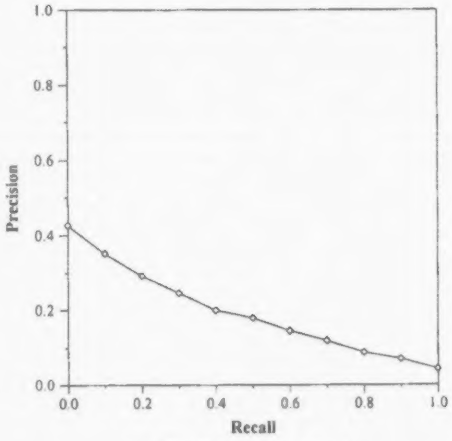
Difference from Median in Average Precision per Topic

Main web track results — Justsystem Corporation

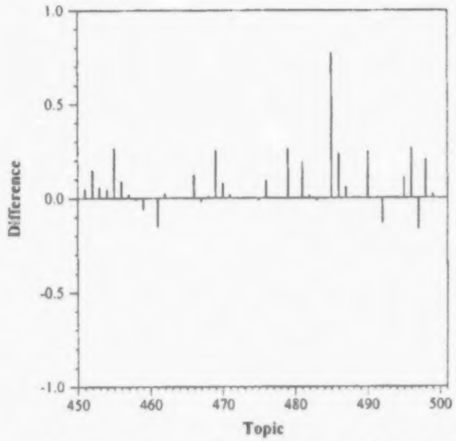
Summary Statistics	
Run Number	jscbt9wls2
Run Description	Automatic, content-link, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1473

Recall Level Precision Averages	
Recall	Precision
0.00	0.4264
0.10	0.3525
0.20	0.2922
0.30	0.2457
0.40	0.1994
0.50	0.1792
0.60	0.1447
0.70	0.1199
0.80	0.0889
0.90	0.0711
1.00	0.0444
Average precision over all relevant docs	
non-interpolated	0.1838

Document Level Averages	
	Precision
At 5 docs	0.2480
At 10 docs	0.2240
At 15 docs	0.1947
At 20 docs	0.1760
At 30 docs	0.1633
At 100 docs	0.1148
At 200 docs	0.0880
At 500 docs	0.0497
At 1000 docs	0.0295
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2027



Recall-Precision Curve



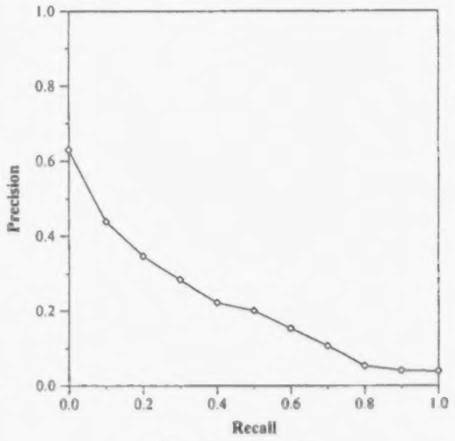
Difference from Median in Average Precision per Topic

Main web track results — NeurOK, LLC

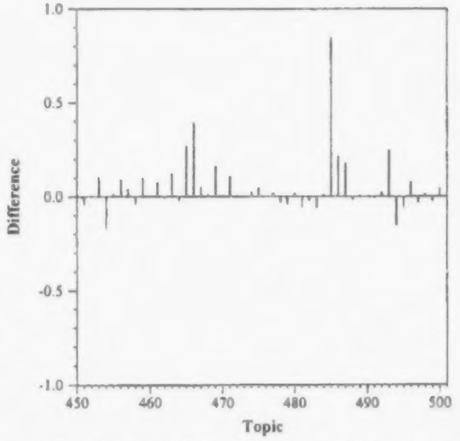
Summary Statistics	
Run Number	NRKlm
Run Description	Automatic, content-only, title+desc+narr
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49772
Relevant:	2617
Rel-ret:	1368

Recall Level Precision Averages	
Recall	Precision
0.00	0.6294
0.10	0.4383
0.20	0.3463
0.30	0.2844
0.40	0.2221
0.50	0.2010
0.60	0.1524
0.70	0.1061
0.80	0.0536
0.90	0.0404
1.00	0.0386
Average precision over all relevant docs	
non-interpolated	0.2064

Document Level Averages	
	Precision
At 5 docs	0.3400
At 10 docs	0.2820
At 15 docs	0.2560
At 20 docs	0.2340
At 30 docs	0.2093
At 100 docs	0.1282
At 200 docs	0.0877
At 500 docs	0.0467
At 1000 docs	0.0274
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2428



Recall-Precision Curve

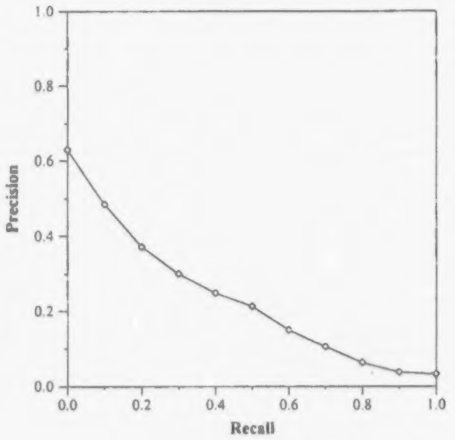


Difference from Median in Average Precision per Topic

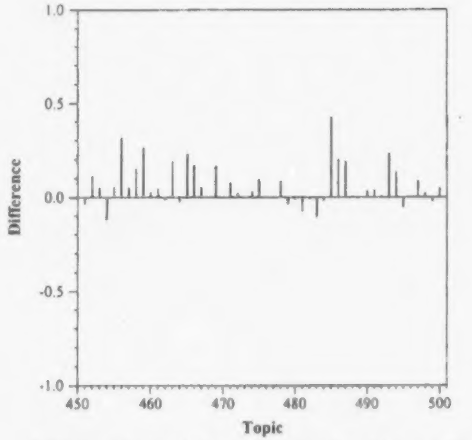
Summary Statistics	
Run Number	NRKprf20
Run Description	Automatic, content-only, title+desc+narr
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49711
Relevant:	2617
Rel-ret:	1548

Recall Level Precision Averages	
Recall	Precision
0.00	0.6298
0.10	0.4852
0.20	0.3711
0.30	0.3013
0.40	0.2500
0.50	0.2141
0.60	0.1496
0.70	0.1049
0.80	0.0639
0.90	0.0378
1.00	0.0322
Average precision over all relevant docs	
non-interpolated	0.2173

Document Level Averages	
	Precision
At 5 docs	0.3600
At 10 docs	0.3260
At 15 docs	0.2933
At 20 docs	0.2740
At 30 docs	0.2360
At 100 docs	0.1444
At 200 docs	0.1009
At 500 docs	0.0538
At 1000 docs	0.0310
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2509



Recall-Precision Curve



Difference from Median in Average Precision per Topic

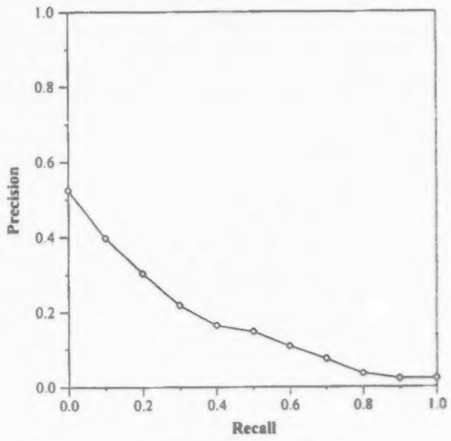
Main web track results — NeurOK, LLC

Summary Statistics	
Run Number	NRKse20
Run Description	Automatic, content-only, title+desc+narr
Number of Topics	50

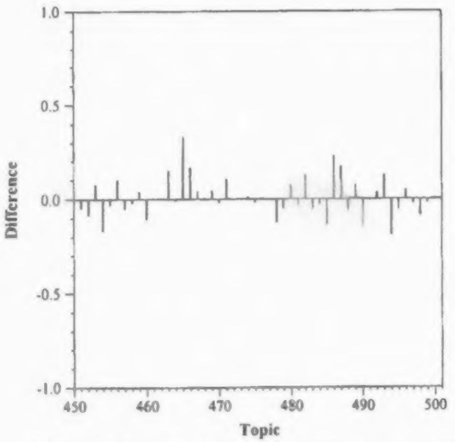
Total number of documents over all topics	
Retrieved:	49620
Relevant:	2617
Rel-ret:	1283

Recall Level Precision Averages	
Recall	Precision
0.00	0.5255
0.10	0.3977
0.20	0.3036
0.30	0.2180
0.40	0.1643
0.50	0.1474
0.60	0.1083
0.70	0.0731
0.80	0.0349
0.90	0.0217
1.00	0.0216
Average precision over all relevant docs	
non-interpolated	0.1642

Document Level Averages	
	Precision
At 5 docs	0.2960
At 10 docs	0.2340
At 15 docs	0.2080
At 20 docs	0.1890
At 30 docs	0.1640
At 100 docs	0.1024
At 200 docs	0.0737
At 500 docs	0.0419
At 1000 docs	0.0257
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2054



Recall-Precision Curve



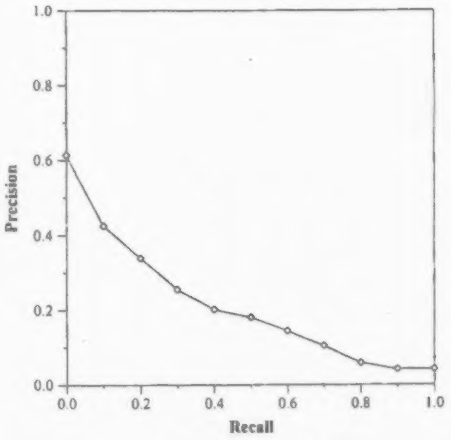
Difference from Median in Average Precision per Topic

Main web track results — NeurOK, LLC

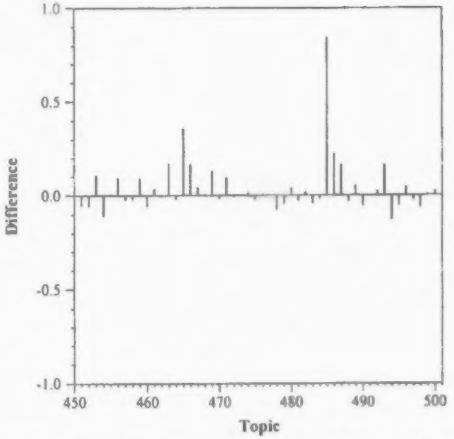
Summary Statistics	
Run Number	NRKse10
Run Description	Automatic, content-only, title+desc+narr
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49707
Relevant:	2617
Rel-ret:	1350

Recall Level Precision Averages	
Recall	Precision
0.00	0.6138
0.10	0.4254
0.20	0.3395
0.30	0.2557
0.40	0.2020
0.50	0.1813
0.60	0.1443
0.70	0.1047
0.80	0.0590
0.90	0.0417
1.00	0.0415
Average precision over all relevant docs	
non-interpolated	0.1960

Document Level Averages	
	Precision
At 5 docs	0.3120
At 10 docs	0.2720
At 15 docs	0.2360
At 20 docs	0.2160
At 30 docs	0.1933
At 100 docs	0.1182
At 200 docs	0.0839
At 500 docs	0.0462
At 1000 docs	0.0270
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2471



Recall-Precision Curve



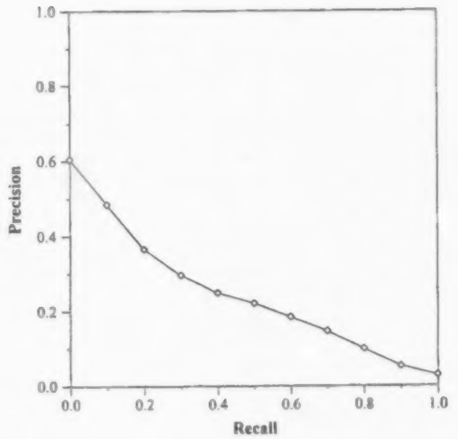
Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	ric9dpx
Run Description	Automatic, content-only, title+desc
Number of Topics	50

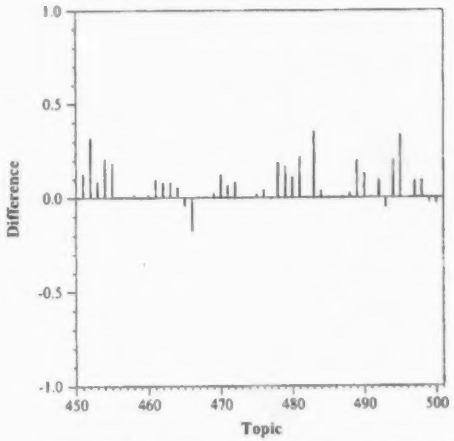
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	2070

Recall Level Precision Averages	
Recall	Precision
0.00	0.6043
0.10	0.4851
0.20	0.3648
0.30	0.2962
0.40	0.2488
0.50	0.2208
0.60	0.1851
0.70	0.1465
0.80	0.0993
0.90	0.0528
1.00	0.0284
Average precision over all relevant docs	
non-interpolated	0.2267

Document Level Averages	
	Precision
At 5 docs	0.3600
At 10 docs	0.3220
At 15 docs	0.2987
At 20 docs	0.2860
At 30 docs	0.2520
At 100 docs	0.1598
At 200 docs	0.1204
At 500 docs	0.0691
At 1000 docs	0.0414
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2418



Recall-Precision Curve



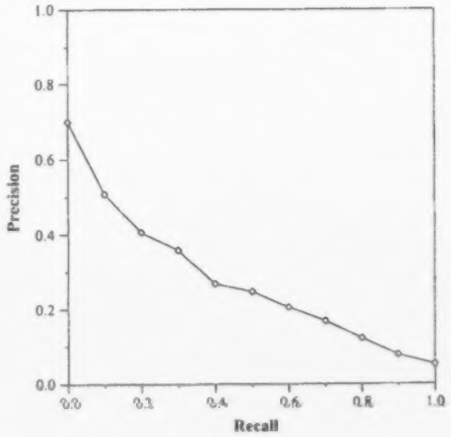
Difference from Median in Average Precision per Topic

Main web track results — RICOH Co., Ltd.

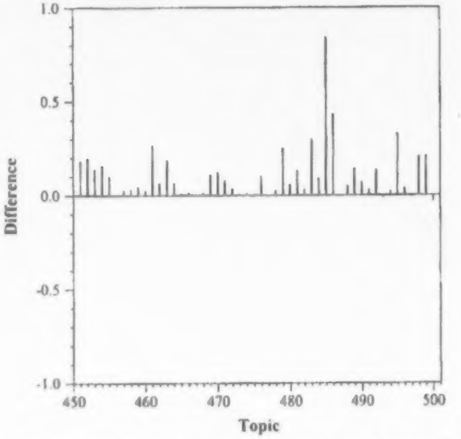
Summary Statistics	
Run Number	ric9dpn
Run Description	Automatic, content-only, title+desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	46838
Relevant:	2617
Rel-ret:	2016

Recall Level Precision Averages	
Recall	Precision
0.00	0.7015
0.10	0.5083
0.20	0.4064
0.30	0.3581
0.40	0.2687
0.50	0.2477
0.60	0.2063
0.70	0.1684
0.80	0.1223
0.90	0.0777
1.00	0.0520
Average precision over all relevant docs	
non-interpolated	0.2616

Document Level Averages	
	Precision
At 5 docs	0.4080
At 10 docs	0.3380
At 15 docs	0.3133
At 20 docs	0.2880
At 30 docs	0.2553
At 100 docs	0.1608
At 200 docs	0.1173
At 500 docs	0.0665
At 1000 docs	0.0403
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2957



Recall-Precision Curve



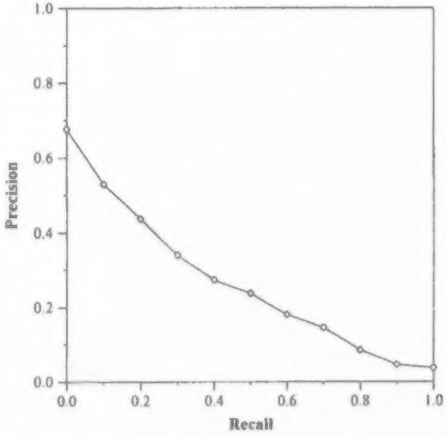
Difference from Median in Average Precision per Topic

Main web track results — Australian National University/CSIRO

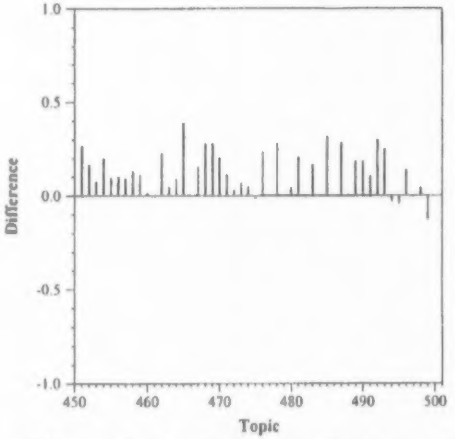
Summary Statistics	
Run Number	acsys9mw0
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	47326
Relevant:	2617
Rel-ret:	1519

Recall Level Precision Averages	
Recall	Precision
0.00	0.6763
0.10	0.5282
0.20	0.4363
0.30	0.3402
0.40	0.2729
0.50	0.2365
0.60	0.1814
0.70	0.1468
0.80	0.0860
0.90	0.0454
1.00	0.0366
Average precision over all relevant docs	
non-interpolated	0.2486

Document Level Averages	
	Precision
At 5 docs	0.4560
At 10 docs	0.3840
At 15 docs	0.3320
At 20 docs	0.3110
At 30 docs	0.2713
At 100 docs	0.1588
At 200 docs	0.0987
At 500 docs	0.0523
At 1000 docs	0.0304
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2910



Recall-Precision Curve

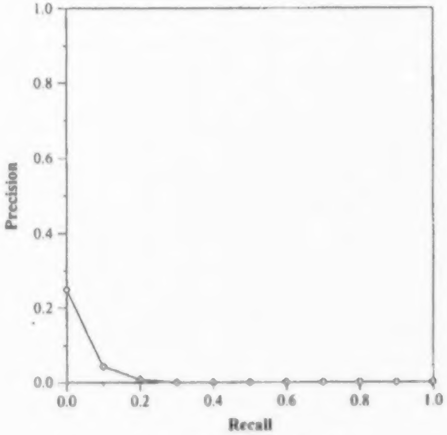


Difference from Median in Average Precision per Topic

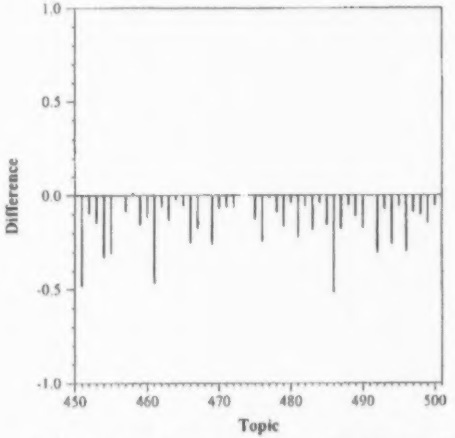
Summary Statistics	
Run Number	Mer9Wtnd
Run Description	Automatic, content-only, title+desc+narr
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	122

Recall Level Precision Averages	
Recall	Precision
0.00	0.2493
0.10	0.0433
0.20	0.0082
0.30	0.0000
0.40	0.0000
0.50	0.0000
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0140

Document Level Averages	
	Precision
At 5 docs	0.1160
At 10 docs	0.0920
At 15 docs	0.0707
At 20 docs	0.0570
At 30 docs	0.0433
At 100 docs	0.0168
At 200 docs	0.0093
At 500 docs	0.0047
At 1000 docs	0.0024
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0295



Recall-Precision Curve



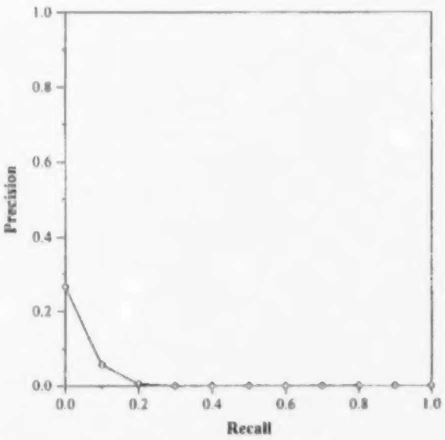
Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	Mer9WtdMr
Run Description	Automatic, content-only, title+desc
Number of Topics	50

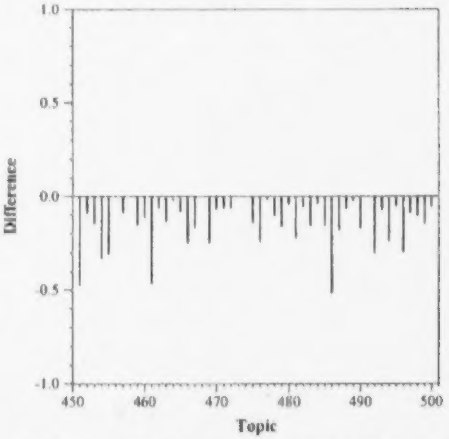
Total number of documents over all topics	
Retrieved:	49720
Relevant:	2617
Rel-ret:	136

Recall Level Precision Averages	
Recall	Precision
0.00	0.2660
0.10	0.0578
0.20	0.0061
0.30	0.0004
0.40	0.0003
0.50	0.0003
0.60	0.0003
0.70	0.0003
0.80	0.0003
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0154

Document Level Averages	
	Precision
At 5 docs	0.1120
At 10 docs	0.0900
At 15 docs	0.0747
At 20 docs	0.0670
At 30 docs	0.0513
At 100 docs	0.0196
At 200 docs	0.0111
At 500 docs	0.0050
At 1000 docs	0.0027
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0307



Recall-Precision Curve



Difference from Median in Average Precision per Topic

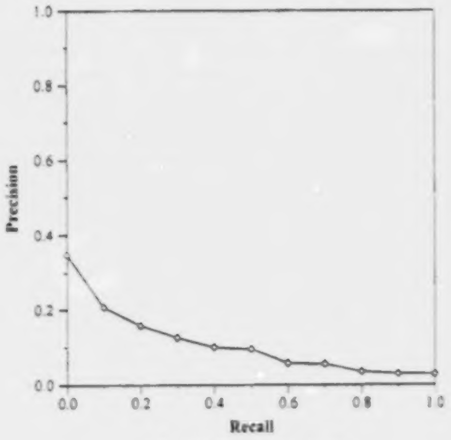
Main web track results — Institut de Recherche en Informatique de Toulouse IRIT/SIG

Summary Statistics	
Run Number	Mer9Wt0
Run Description	Automatic, content-only, title only
Number of Topics	50

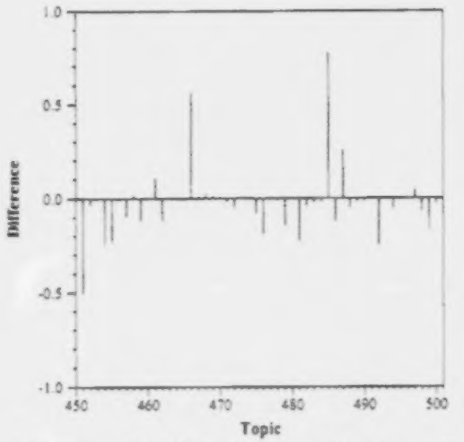
Total number of documents over all topics	
Retrieved:	44604
Relevant:	2617
Rel-ret:	488

Recall Level Precision Averages	
Recall	Precision
0.00	0.3485
0.10	0.2073
0.20	0.1585
0.30	0.1254
0.40	0.1006
0.50	0.0957
0.60	0.0579
0.70	0.0558
0.80	0.0356
0.90	0.0304
1.00	0.0290
Average precision over all relevant docs	
non-interpolated	0.0996

Document Level Averages	
	Precision
At 5 docs	0.1600
At 10 docs	0.1280
At 15 docs	0.1147
At 20 docs	0.1020
At 30 docs	0.0887
At 100 docs	0.0524
At 200 docs	0.0338
At 500 docs	0.0162
At 1000 docs	0.0098
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1274



Recall-Precision Curve

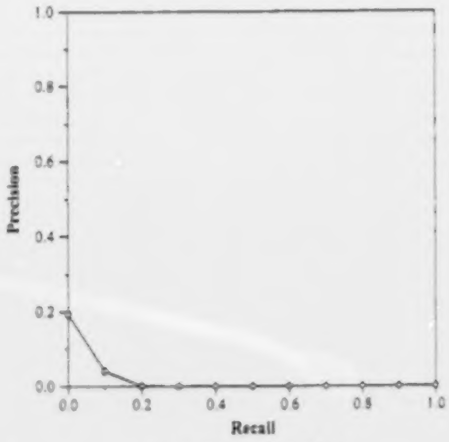


Difference from Median in Average Precision per Topic

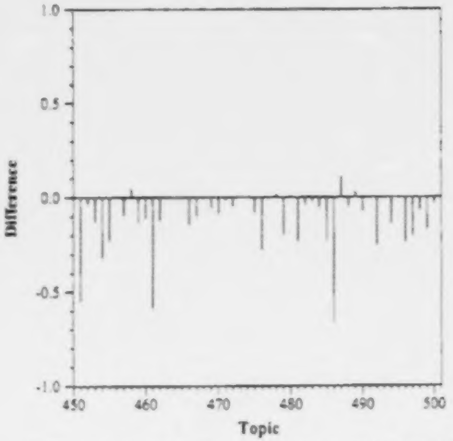
Summary Statistics	
Run Number	Mer9Wt1
Run Description	Automatic, content-only, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	113

Recall Level Precision Averages	
Recall	Precision
0.00	0.1938
0.10	0.0405
0.20	0.0011
0.30	0.0000
0.40	0.0000
0.50	0.0000
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0114

Document Level Averages	
	Precision
At 5 docs	0.0880
At 10 docs	0.0700
At 15 docs	0.0640
At 20 docs	0.0540
At 30 docs	0.0440
At 100 docs	0.0184
At 200 docs	0.0101
At 500 docs	0.0043
At 1000 docs	0.0023
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0242



Recall-Precision Curve

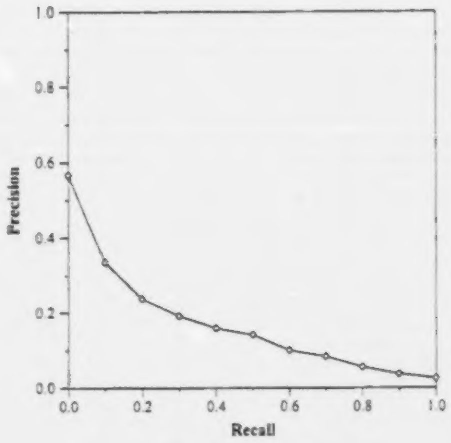


Difference from Median in Average Precision per Topic

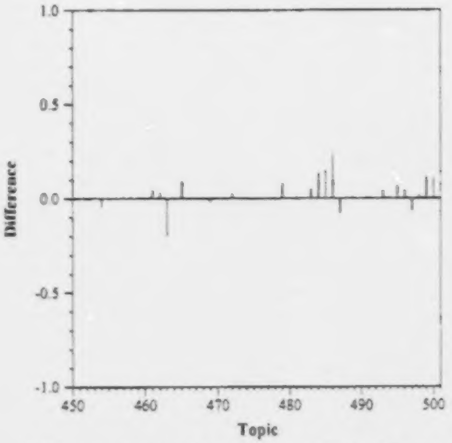
Summary Statistics	
Run Number	dcu00ca
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	41203
Relevant:	2617
Rel-ret:	1037

Recall Level Precision Averages	
Recall	Precision
0.00	0.5662
0.10	0.3345
0.20	0.2385
0.30	0.1918
0.40	0.1588
0.50	0.1420
0.60	0.1003
0.70	0.0834
0.80	0.0551
0.90	0.0366
1.00	0.0251
Average precision over all relevant docs	
non-interpolated	0.1519

Document Level Averages	
	Precision
At 5 docs	0.3120
At 10 docs	0.2780
At 15 docs	0.2440
At 20 docs	0.2160
At 30 docs	0.1800
At 100 docs	0.1076
At 200 docs	0.0686
At 500 docs	0.0330
At 1000 docs	0.0207
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1777



Recall-Precision Curve



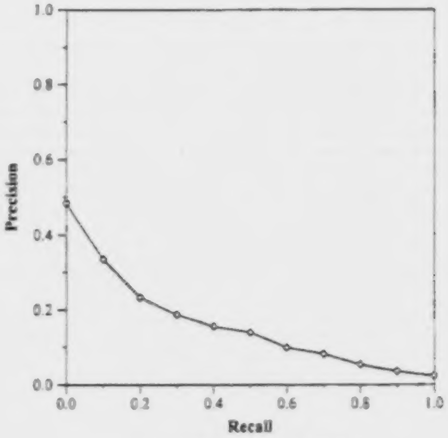
Difference from Median in Average Precision per Topic

Main web track results — Dublin City University

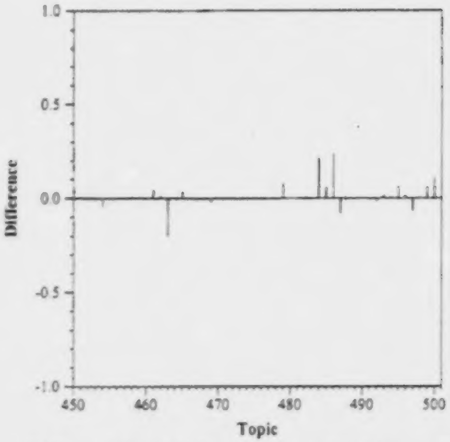
Summary Statistics	
Run Number	dcu00la
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	41203
Relevant:	2617
Rel-ret:	1037

Recall Level Precision Averages	
Recall	Precision
0.00	0.4835
0.10	0.3352
0.20	0.2326
0.30	0.1875
0.40	0.1558
0.50	0.1400
0.60	0.0983
0.70	0.0814
0.80	0.0531
0.90	0.0346
1.00	0.0231
Average precision over all relevant docs	
non-interpolated	0.1450

Document Level Averages	
	Precision
At 5 docs	0.2880
At 10 docs	0.2740
At 15 docs	0.2400
At 20 docs	0.2130
At 30 docs	0.1800
At 100 docs	0.1076
At 200 docs	0.0686
At 500 docs	0.0330
At 1000 docs	0.0207
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1792



Recall-Precision Curve



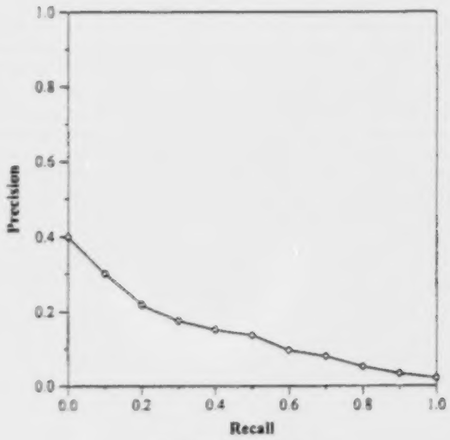
Difference from Median in Average Precision per Topic

Main web track results — Dublin City University

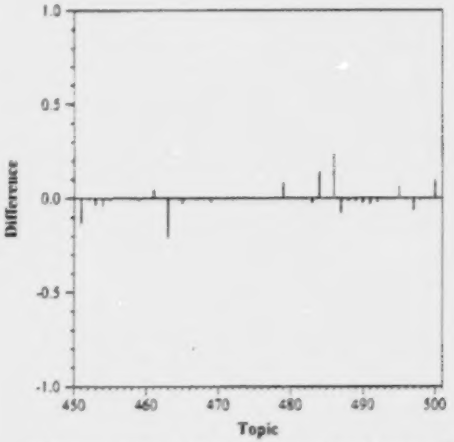
Summary Statistics	
Run Number	dcu00lb
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	41203
Relevant:	2617
Rel-ret:	1037

Recall Level Precision Averages	
Recall	Precision
0.00	0.4001
0.10	0.3016
0.20	0.2182
0.30	0.1755
0.40	0.1515
0.50	0.1369
0.60	0.0959
0.70	0.0800
0.80	0.0517
0.90	0.0332
1.00	0.0217
Average precision over all relevant docs	
non-interpolated	0.1324

Document Level Averages	
	Precision
At 5 docs	0.2280
At 10 docs	0.2440
At 15 docs	0.2240
At 20 docs	0.2070
At 30 docs	0.1767
At 100 docs	0.1076
At 200 docs	0.0686
At 500 docs	0.0330
At 1000 docs	0.0207
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1707



Recall-Precision Curve



Difference from Median in Average Precision per Topic

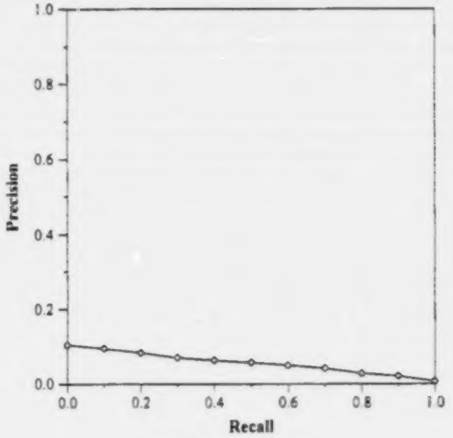
Main web track results — TwentyOne

Summary Statistics	
Run Number	tnout9t2lk50
Run Description	Automatic, content-link, title only
Number of Topics	50

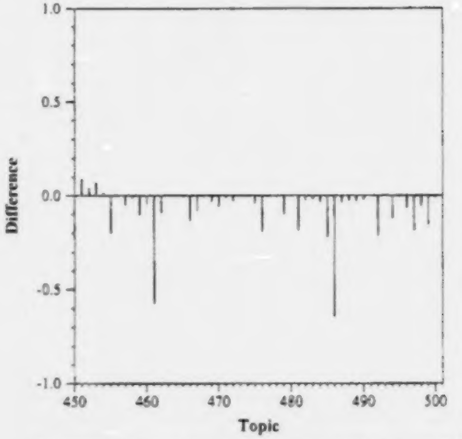
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1284

Recall Level Precision Averages	
Recall	Precision
0.00	0.1060
0.10	0.0962
0.20	0.0845
0.30	0.0709
0.40	0.0638
0.50	0.0577
0.60	0.0504
0.70	0.0423
0.80	0.0288
0.90	0.0213
1.00	0.0068
Average precision over all relevant docs	
non-interpolated	0.0488

Document Level Averages	
	Precision
At 5 docs	0.0320
At 10 docs	0.0320
At 15 docs	0.0360
At 20 docs	0.0410
At 30 docs	0.0407
At 100 docs	0.0286
At 200 docs	0.0320
At 500 docs	0.0410
At 1000 docs	0.0257
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0384



Recall-Precision Curve



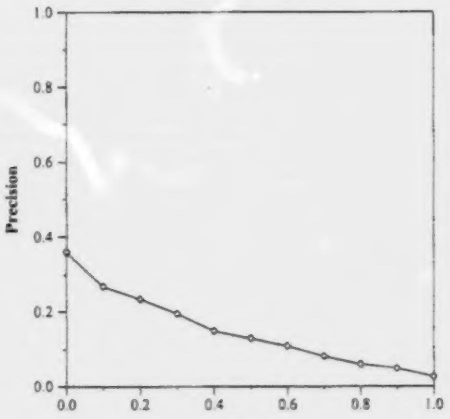
Difference from Median in Average Precision per Topic

Main web track results — TwentyOne

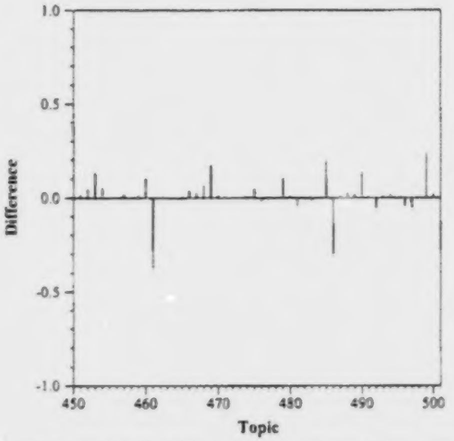
Summary Statistics	
Run Number	tnout9t2lc50
Run Description	Automatic, content-link, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	45926
Relevant:	2617
Rel-ret:	1344

Recall Level Precision Averages	
Recall	Precision
0.00	0.3615
0.10	0.2683
0.20	0.2342
0.30	0.1946
0.40	0.1487
0.50	0.1296
0.60	0.1093
0.70	0.0823
0.80	0.0606
0.90	0.0498
1.00	0.0266
Average precision over all relevant docs	
non-interpolated	0.1337

Document Level Averages	
	Precision
At 5 docs	0.2000
At 10 docs	0.1980
At 15 docs	0.1933
At 20 docs	0.1730
At 30 docs	0.1560
At 100 docs	0.1074
At 200 docs	0.0772
At 500 docs	0.0451
At 1000 docs	0.0269
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1614



Recall-Precision Curve



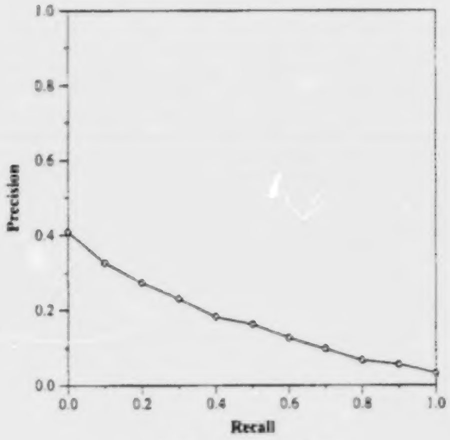
Difference from Median in Average Precision per Topic

Main web track results — TwentyOne

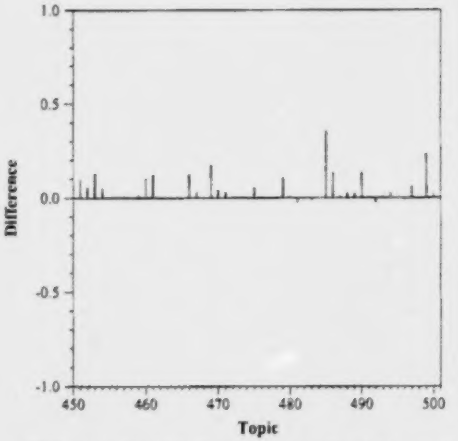
Summary Statistics	
Run Number	tnout9t2lc10
Run Description	Automatic, content-link, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	45865
Relevant:	2617
Rel-ret:	1344

Recall Level Precision Averages	
Recall	Precision
0.00	0.4089
0.10	0.3281
0.20	0.2746
0.30	0.2314
0.40	0.1819
0.50	0.1623
0.60	0.1255
0.70	0.0977
0.80	0.0674
0.90	0.0566
1.00	0.0334
Average precision over all relevant docs	
non-interpolated	0.1630

Document Level Averages	
	Precision
At 5 docs	0.2400
At 10 docs	0.2140
At 15 docs	0.1933
At 20 docs	0.1720
At 30 docs	0.1573
At 100 docs	0.1074
At 200 docs	0.0770
At 500 docs	0.0450
At 1000 docs	0.0269
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2036



Recall-Precision Curve



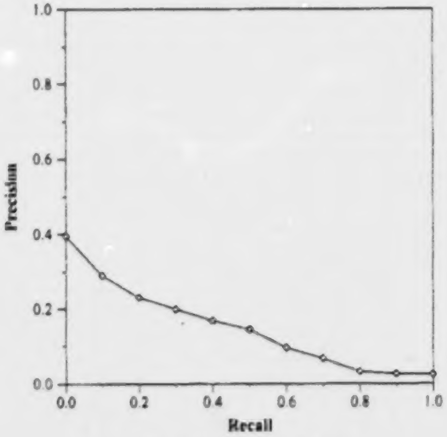
Difference from Median in Average Precision per Topic

Main web track results — Fujitsu Laboratories Ltd.

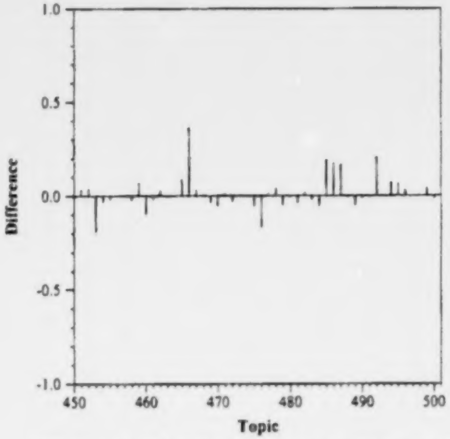
Summary Statistics	
Run Number	Flab9atN
Run Description	Automatic, content-only, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1179

Recall Level Precision Averages	
Recall	Precision
0.00	0.3954
0.10	0.2902
0.20	0.2314
0.30	0.1993
0.40	0.1684
0.50	0.1439
0.60	0.0949
0.70	0.0680
0.80	0.0325
0.90	0.0260
1.00	0.0244
Average precision over all relevant docs	
non-interpolated	0.1360

Document Level Averages	
	Precision
At 5 docs	0.2160
At 10 docs	0.1940
At 15 docs	0.1787
At 20 docs	0.1570
At 30 docs	0.1367
At 100 docs	0.0912
At 200 docs	0.0637
At 500 docs	0.0366
At 1000 docs	0.0236
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1534



Recall-Precision Curve



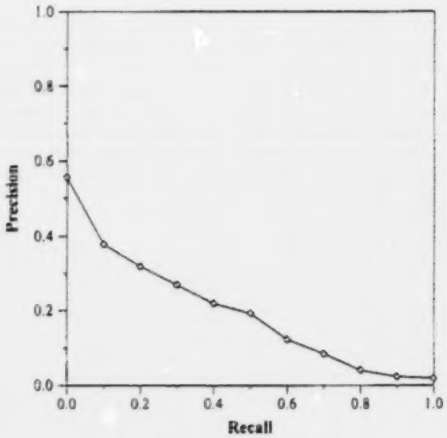
Difference from Median in Average Precision per Topic

Main web track results — Fujitsu Laboratories Ltd.

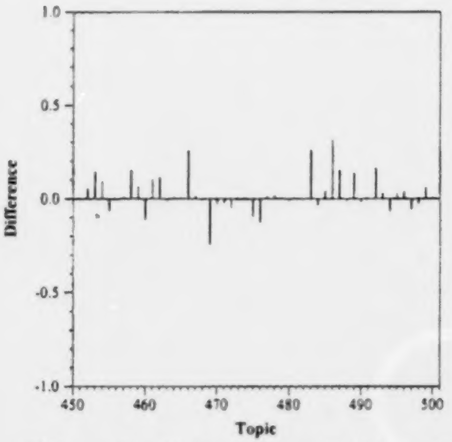
Summary Statistics	
Run Number	Flab9atdN
Run Description	Automatic, content-only, title+desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1526

Recall Level Precision Averages	
Recall	Precision
0.00	0.5568
0.10	0.3791
0.20	0.3189
0.30	0.2692
0.40	0.2194
0.50	0.1926
0.60	0.1223
0.70	0.0844
0.80	0.0416
0.90	0.0241
1.00	0.0194
Average precision over all relevant docs	
non-interpolated	0.1816

Document Level Averages	
	Precision
At 5 docs	0.3400
At 10 docs	0.2980
At 15 docs	0.2480
At 20 docs	0.2320
At 30 docs	0.1993
At 100 docs	0.1282
At 200 docs	0.0894
At 500 docs	0.0490
At 1000 docs	0.0305
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2079



Recall-Precision Curve



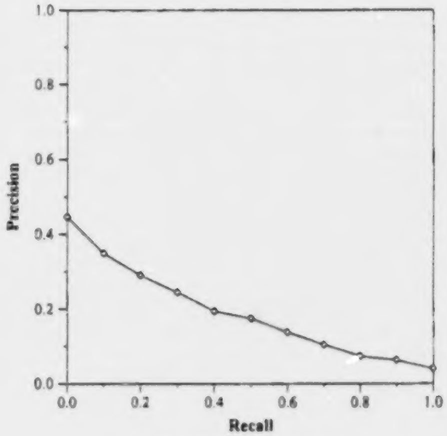
Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	tnout9t2
Run Description	Automatic, content-only, title only
Number of Topics	50

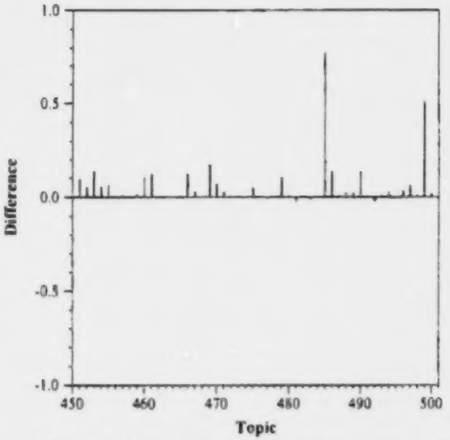
Total number of documents over all topics	
Retrieved:	45826
Relevant:	2617
Rel-ret:	1344

Recall Level Precision Averages	
Recall	Precision
0.00	0.4470
0.10	0.3490
0.20	0.2909
0.30	0.2448
0.40	0.1952
0.50	0.1756
0.60	0.1388
0.70	0.1044
0.80	0.0740
0.90	0.0632
1.00	0.0401
Average precision over all relevant docs	
non-interpolated	0.1801

Document Level Averages	
	Precision
At 5 docs	0.2520
At 10 docs	0.2140
At 15 docs	0.1933
At 20 docs	0.1740
At 30 docs	0.1587
At 100 docs	0.1074
At 200 docs	0.0770
At 500 docs	0.0450
At 1000 docs	0.0269
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2169



Recall-Precision Curve



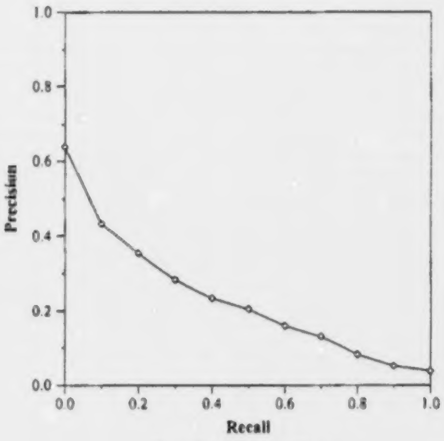
Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	tnout9f1
Run Description	Automatic, content-only, title+desc+narr
Number of Topics	50

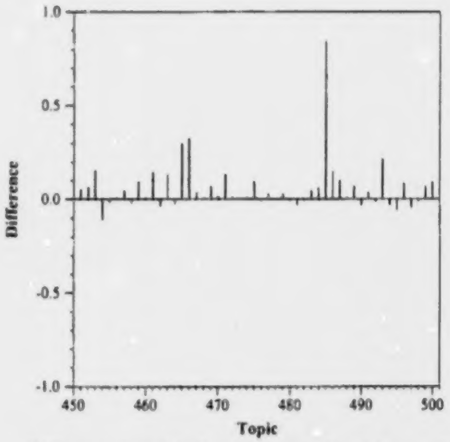
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1561

Recall Level Precision Averages	
Recall	Precision
0.00	0.6388
0.10	0.4343
0.20	0.3540
0.30	0.2835
0.40	0.2344
0.50	0.2061
0.60	0.1595
0.70	0.1311
0.80	0.0826
0.90	0.0511
1.00	0.0373
Average precision over all relevant docs	
non-interpolated	0.2178

Document Level Averages	
	Precision
At 5 docs	0.3600
At 10 docs	0.2900
At 15 docs	0.2680
At 20 docs	0.2460
At 30 docs	0.2067
At 100 docs	0.1296
At 200 docs	0.0971
At 500 docs	0.0527
At 1000 docs	0.0312
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2500



Recall-Precision Curve



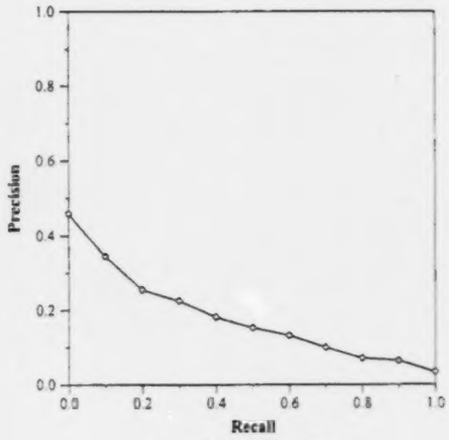
Difference from Median in Average Precision per Topic

Main web track results — University of Waterloo MultiText project

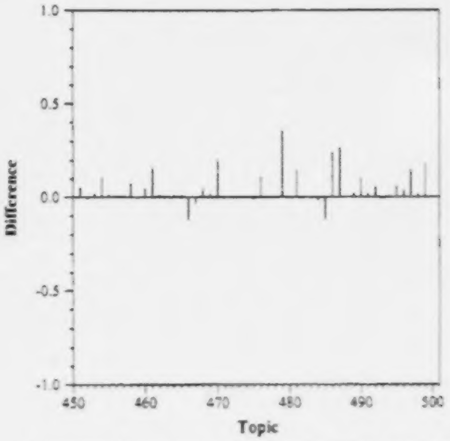
Summary Statistics	
Run Number	uwmt9w10g0
Run Description	Automatic, content-only, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	45636
Relevant:	2617
Rel-ret:	1325

Recall Level Precision Averages	
Recall	Precision
0.00	0.4601
0.10	0.3443
0.20	0.2552
0.30	0.2252
0.40	0.1828
0.50	0.1537
0.60	0.1329
0.70	0.1006
0.80	0.0709
0.90	0.0642
1.00	0.0347
Average precision over all relevant docs	
non-interpolated	0.1654

Document Level Averages	
	Precision
At 5 docs	0.2840
At 10 docs	0.2380
At 15 docs	0.2080
At 20 docs	0.1970
At 30 docs	0.1633
At 100 docs	0.1100
At 200 docs	0.0847
At 500 docs	0.0452
At 1000 docs	0.0265
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1883



Recall-Precision Curve



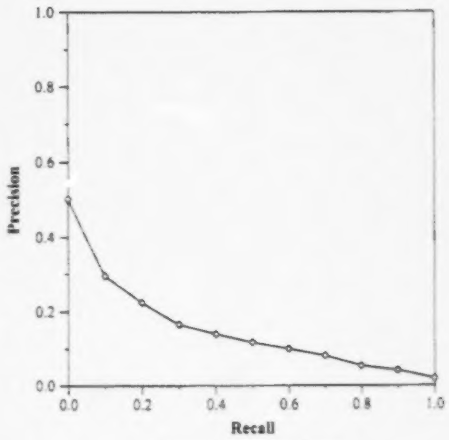
Difference from Median in Average Precision per Topic

Main web track results — University of Waterloo MultiText project

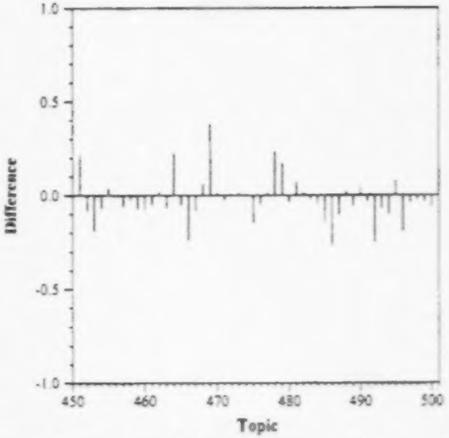
Summary Statistics	
Run Number	uwmt9w10g1
Run Description	Automatic, content-only, title+desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49870
Relevant:	2617
Rel-ret:	1241

Recall Level Precision Averages	
Recall	Precision
0.00	0.5000
0.10	0.2949
0.20	0.2230
0.30	0.1663
0.40	0.1407
0.50	0.1180
0.60	0.0999
0.70	0.0818
0.80	0.0534
0.90	0.0414
1.00	0.0195
Average precision over all relevant docs	
non-interpolated	0.1331

Document Level Averages	
	Precision
At 5 docs	0.2840
At 10 docs	0.2600
At 15 docs	0.2387
At 20 docs	0.2160
At 30 docs	0.1727
At 100 docs	0.0950
At 200 docs	0.0692
At 500 docs	0.0442
At 1000 docs	0.0248
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1559



Recall-Precision Curve



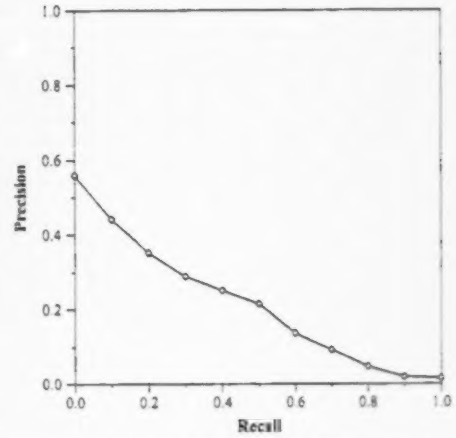
Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	Sab9web5
Run Description	Automatic, content-link, title+desc+narr
Number of Topics	50

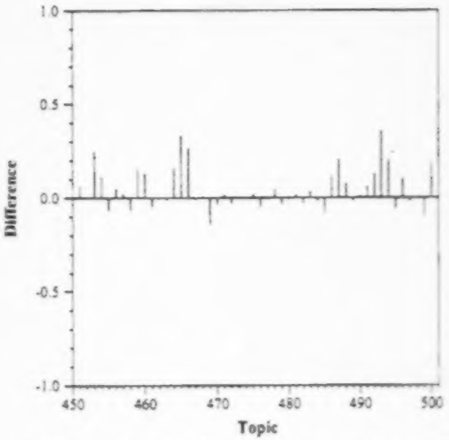
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1468

Recall Level Precision Averages	
Recall	Precision
0.00	0.5604
0.10	0.4411
0.20	0.3535
0.30	0.2898
0.40	0.2514
0.50	0.2152
0.60	0.1363
0.70	0.0926
0.80	0.0478
0.90	0.0197
1.00	0.0156
Average precision over all relevant docs	
non-interpolated	0.2018

Document Level Averages	
	Precision
At 5 docs	0.3280
At 10 docs	0.3140
At 15 docs	0.3053
At 20 docs	0.2940
At 30 docs	0.2553
At 100 docs	0.1464
At 200 docs	0.0952
At 500 docs	0.0504
At 1000 docs	0.0294
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2400



Recall-Precision Curve



Difference from Median in Average Precision per Topic

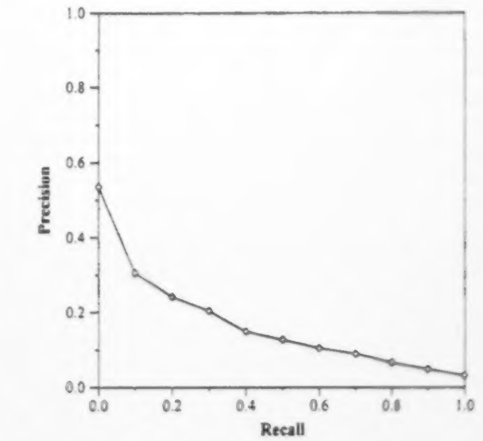
Main web track results — State University of New York at Buffalo

Summary Statistics	
Run Number	xvsmmain
Run Description	Automatic, content-only, title+desc
Number of Topics	50

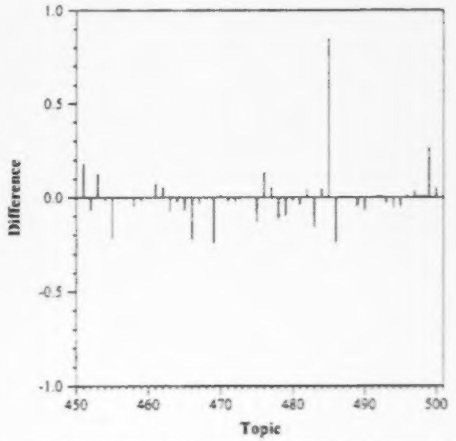
Total number of documents over all topics	
Retrieved:	35309
Relevant:	2617
Rel-ret:	1238

Recall Level Precision Averages	
Recall	Precision
0.00	0.5356
0.10	0.3055
0.20	0.2427
0.30	0.2038
0.40	0.1488
0.50	0.1281
0.60	0.1051
0.70	0.0897
0.80	0.0668
0.90	0.0497
1.00	0.0322
Average precision over all relevant docs	
non-interpolated	0.1521

Document Level Averages	
	Precision
At 5 docs	0.2240
At 10 docs	0.2140
At 15 docs	0.1933
At 20 docs	0.1780
At 30 docs	0.1580
At 100 docs	0.1020
At 200 docs	0.0738
At 500 docs	0.0417
At 1000 docs	0.0248
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1780



Recall-Precision Curve



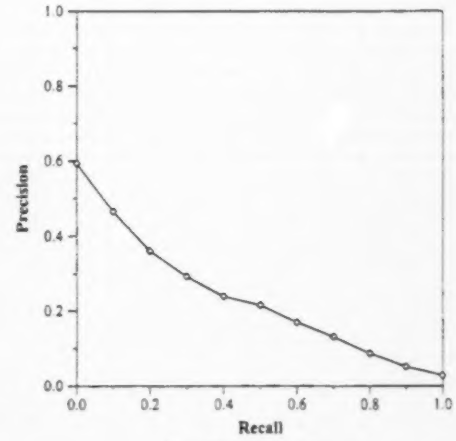
Difference from Median in Average Precision per Topic

Main web track results — RICOH Co., Ltd.

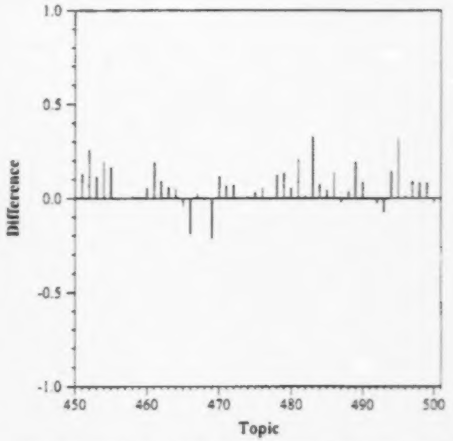
Summary Statistics	
Run Number	ric9dsx
Run Description	Automatic, content-only, title+desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1997

Recall Level Precision Averages	
Recall	Precision
0.00	0.5941
0.10	0.4650
0.20	0.3604
0.30	0.2930
0.40	0.2391
0.50	0.2152
0.60	0.1692
0.70	0.1316
0.80	0.0867
0.90	0.0516
1.00	0.0282
Average precision over all relevant docs	
non-interpolated	0.2201

Document Level Averages	
	Precision
At 5 docs	0.3520
At 10 docs	0.3240
At 15 docs	0.3000
At 20 docs	0.2750
At 30 docs	0.2420
At 100 docs	0.1556
At 200 docs	0.1149
At 500 docs	0.0659
At 1000 docs	0.0399
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2265



Recall-Precision Curve



Difference from Median in Average Precision per Topic

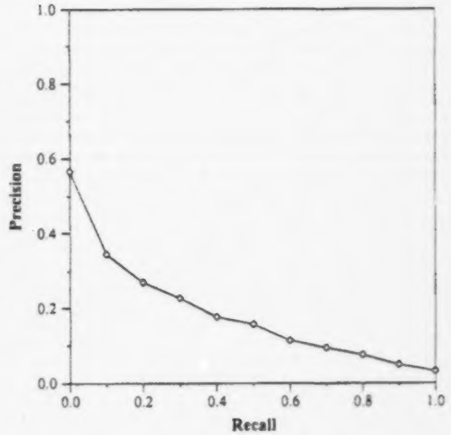
Main web track results — State University of New York at Buffalo

Summary Statistics	
Run Number	xvsmtdn
Run Description	Automatic, content-only, title+desc+narr
Number of Topics	50

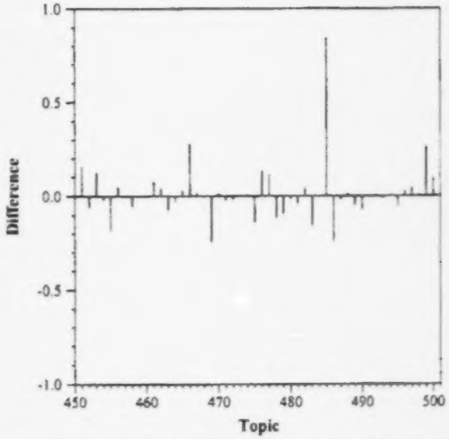
Total number of documents over all topics	
Retrieved:	35477
Relevant:	2617
Rel-ret:	1260

Recall Level Precision Averages	
Recall	Precision
0.00	0.5663
0.10	0.3459
0.20	0.2712
0.30	0.2277
0.40	0.1765
0.50	0.1567
0.60	0.1128
0.70	0.0939
0.80	0.0755
0.90	0.0502
1.00	0.0325
Average precision over all relevant docs	
non-interpolated	0.1694

Document Level Averages	
	Precision
At 5 docs	0.2680
At 10 docs	0.2380
At 15 docs	0.2107
At 20 docs	0.1930
At 30 docs	0.1660
At 100 docs	0.1044
At 200 docs	0.0748
At 500 docs	0.0437
At 1000 docs	0.0252
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1981



Recall-Precision Curve



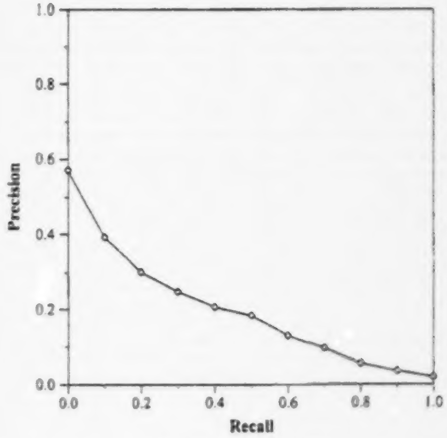
Difference from Median in Average Precision per Topic

Main web track results — Johns Hopkins University, APL

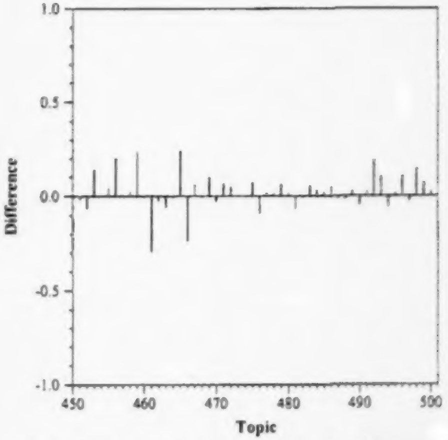
Summary Statistics	
Run Number	apl9tdn
Run Description	Automatic, content-only, title+desc+narr
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1584

Recall Level Precision Averages	
Recall	Precision
0.00	0.5704
0.10	0.3921
0.20	0.3008
0.30	0.2481
0.40	0.2062
0.50	0.1836
0.60	0.1297
0.70	0.0961
0.80	0.0554
0.90	0.0356
1.00	0.0198
Average precision over all relevant docs	
non-interpolated	0.1785

Document Level Averages	
	Precision
At 5 docs	0.3440
At 10 docs	0.2860
At 15 docs	0.2600
At 20 docs	0.2300
At 30 docs	0.2027
At 100 docs	0.1334
At 200 docs	0.0928
At 500 docs	0.0527
At 1000 docs	0.0317
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2059



Recall-Precision Curve



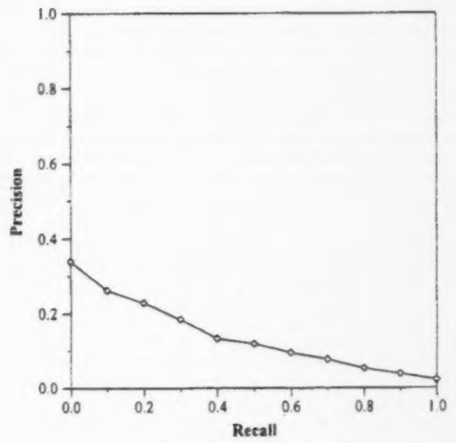
Difference from Median in Average Precision per Topic

Main web track results — Johns Hopkins University, APL

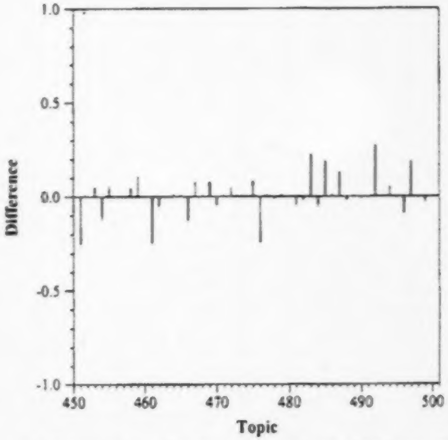
Summary Statistics	
Run Number	apl9t
Run Description	Automatic, content-only, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1276

Recall Level Precision Averages	
Recall	Precision
0.00	0.3398
0.10	0.2618
0.20	0.2284
0.30	0.1851
0.40	0.1339
0.50	0.1197
0.60	0.0949
0.70	0.0775
0.80	0.0522
0.90	0.0371
1.00	0.0210
Average precision over all relevant docs	
non-interpolated	0.1272

Document Level Averages	
	Precision
At 5 docs	0.1440
At 10 docs	0.1340
At 15 docs	0.1307
At 20 docs	0.1260
At 30 docs	0.1293
At 100 docs	0.0864
At 200 docs	0.0658
At 500 docs	0.0412
At 1000 docs	0.0255
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1466



Recall-Precision Curve



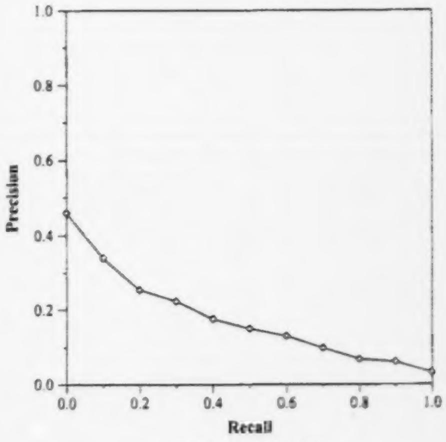
Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	uwmt9w10g2
Run Description	Automatic, content-link, title only
Number of Topics	50

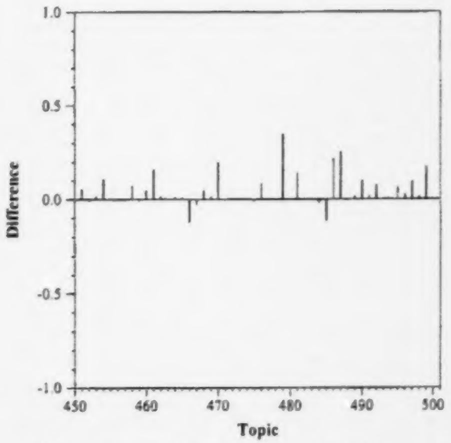
Total number of documents over all topics	
Retrieved:	45636
Relevant:	2617
Rel-ret:	1325

Recall Level Precision Averages	
Recall	Precision
0.00	0.4594
0.10	0.3407
0.20	0.2532
0.30	0.2232
0.40	0.1766
0.50	0.1515
0.60	0.1312
0.70	0.0996
0.80	0.0681
0.90	0.0617
1.00	0.0328
Average precision over all relevant docs	
non-interpolated	0.1631

Document Level Averages	
	Precision
At 5 docs	0.2840
At 10 docs	0.2360
At 15 docs	0.2053
At 20 docs	0.1920
At 30 docs	0.1607
At 100 docs	0.1078
At 200 docs	0.0851
At 500 docs	0.0455
At 1000 docs	0.0265
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1833



Recall-Precision Curve

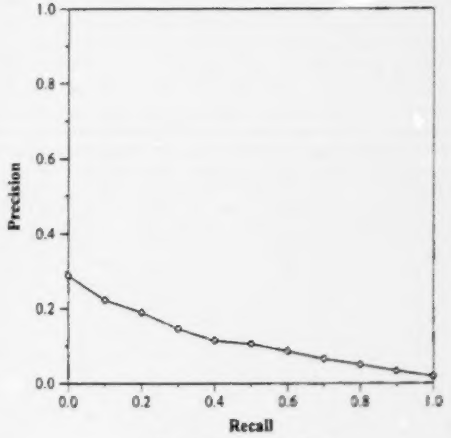


Difference from Median in Average Precision per Topic

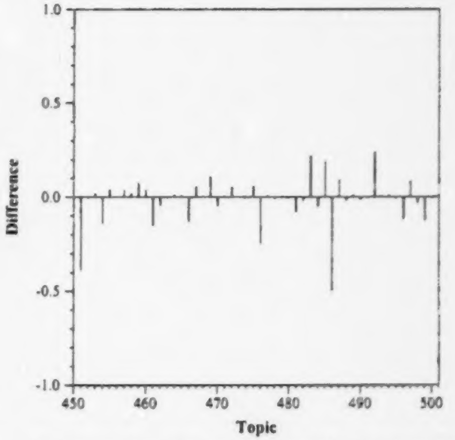
Summary Statistics	
Run Number	apl9lt
Run Description	Automatic, content-link, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1276

Recall Level Precision Averages	
Recall	Precision
0.00	0.2893
0.10	0.2226
0.20	0.1898
0.30	0.1466
0.40	0.1150
0.50	0.1062
0.60	0.0862
0.70	0.0662
0.80	0.0507
0.90	0.0343
1.00	0.0190
Average precision over all relevant docs	
non-interpolated	0.1062

Document Level Averages	
	Precision
At 5 docs	0.1400
At 10 docs	0.1160
At 15 docs	0.1107
At 20 docs	0.1090
At 30 docs	0.1120
At 100 docs	0.0832
At 200 docs	0.0639
At 500 docs	0.0410
At 1000 docs	0.0255
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1186



Recall-Precision Curve

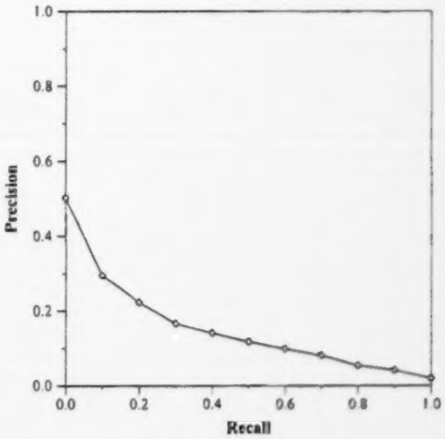


Difference from Median in Average Precision per Topic

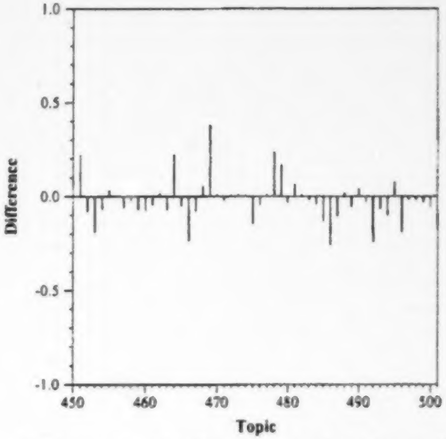
Summary Statistics	
Run Number	uwmt9w10g3
Run Description	Automatic, content-link, title+desc
Number of Topics	50
Total number of doc[illegible]ents over all topics	
Retrieved:	49870
Relevant:	2617
Rel-ret:	1241

Recall Level Precision Averages	
Recall	Precision
0.00	0.5013
0.10	0.2956
0.20	0.2231
0.30	0.1667
0.40	0.1402
0.50	0.1175
0.60	0.0995
0.70	0.0818
0.80	0.0542
0.90	0.0423
1.00	0.0198
Average precision over all relevant docs	
non-interpolated	0.1336

Document Level Averages	
	Precision
At 5 docs	0.2840
At 10 docs	0.2620
At 15 docs	0.2400
At 20 docs	0.2110
At 30 docs	0.1740
At 100 docs	0.0956
At 200 docs	0.0700
At 500 docs	0.0440
At 1000 docs	0.0248
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1562



Recall-Precision Curve

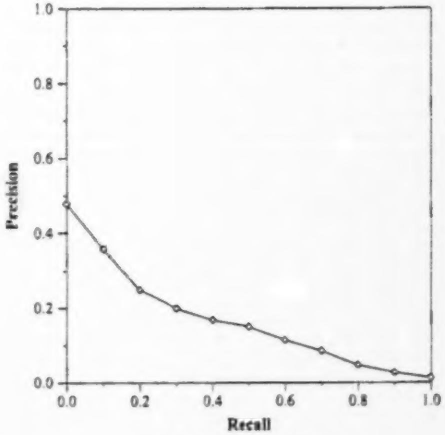


Difference from Median in Average Precision per Topic

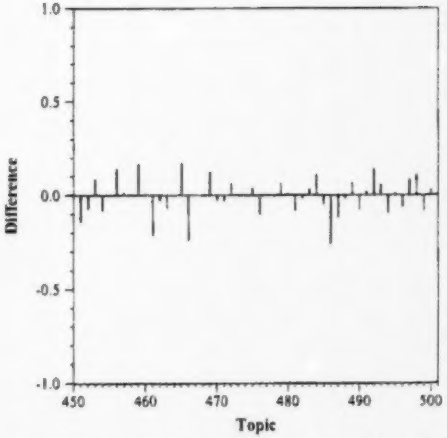
Summary Statistics	
Run Number	apl9ltdn
Run Description	Automatic, content-link, title+desc+narr
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1584

Recall Level Precision Averages	
Recall	Precision
0.00	0.4775
0.10	0.3578
0.20	0.2502
0.30	0.1990
0.40	0.1674
0.50	0.1490
0.60	0.1136
0.70	0.0849
0.80	0.0482
0.90	0.0272
1.00	0.0131
Average precision over all relevant docs	
non-interpolated	0.1494

Document Level Averages	
	Precision
At 5 docs	0.2480
At 10 docs	0.2320
At 15 docs	0.2240
At 20 docs	0.2050
At 30 docs	0.1793
At 100 docs	0.1242
At 200 docs	0.0911
At 500 docs	0.0520
At 1000 docs	0.0317
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1785



Recall-Precision Curve

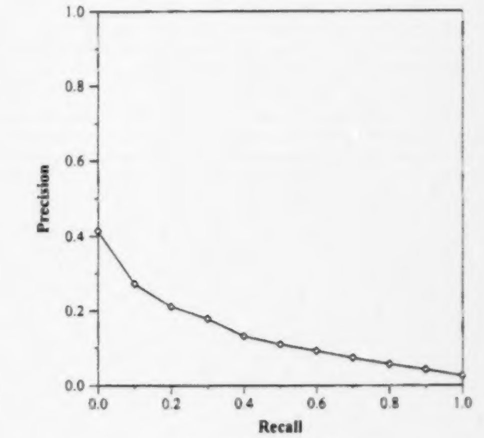


Difference from Median in Average Precision per Topic

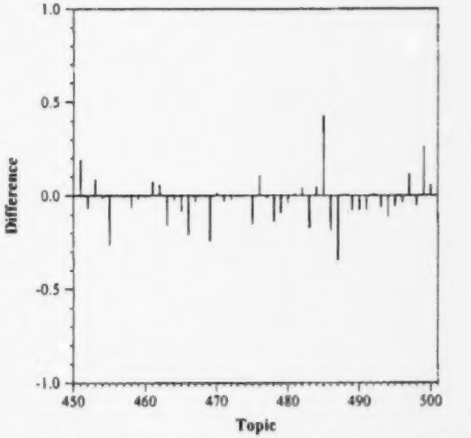
Summary Statistics	
Run Number	xvsmtitle
Run Description	Automatic, content-only, title+narr
Number of Topics	50
Total number of documents over all topics	
Retrieved:	32318
Relevant:	2617
Rel-ret:	1067

Recall Level Precision Averages	
Recall	Precision
0.00	0.4141
0.10	0.2728
0.20	0.2122
0.30	0.1789
0.40	0.1308
0.50	0.1082
0.60	0.0906
0.70	0.0723
0.80	0.0559
0.90	0.0421
1.00	0.0256
Average precision over all relevant docs	
non-interpolated	0.1278

Document Level Averages	
	Precision
At 5 docs	0.1960
At 10 docs	0.1840
At 15 docs	0.1693
At 20 docs	0.1590
At 30 docs	0.1387
At 100 docs	0.0888
At 200 docs	0.0644
At 500 docs	0.0365
At 1000 docs	0.0213
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1539



Recall-Precision Curve



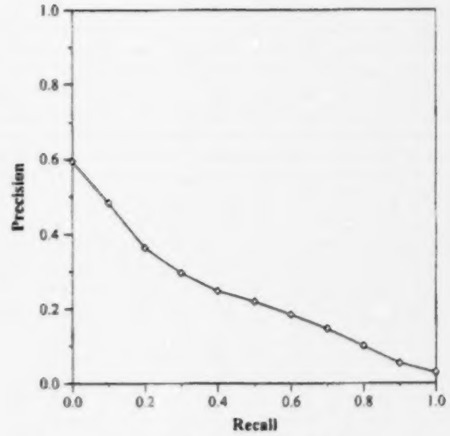
Difference from Median in Average Precision per Topic

Main web track results — RICOH Co., Ltd.

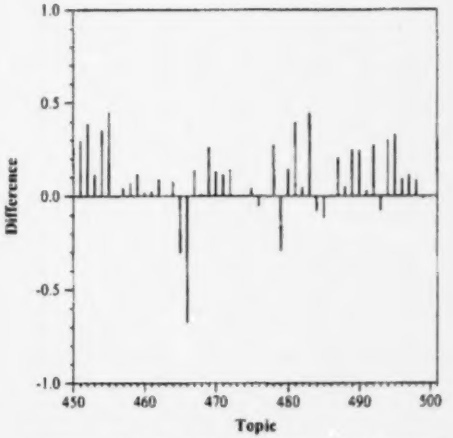
Summary Statistics	
Run Number	ric9dpxL
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	2070

Recall Level Precision Averages	
Recall	Precision
0.00	0.5943
0.10	0.4840
0.20	0.3637
0.30	0.2953
0.40	0.2479
0.50	0.2201
0.60	0.1847
0.70	0.1463
0.80	0.0992
0.90	0.0528
1.00	0.0284
Average precision over all relevant docs	
non-interpolated	0.2257

Document Level Averages	
	Precision
At 5 docs	0.3600
At 10 docs	0.3160
At 15 docs	0.2947
At 20 docs	0.2840
At 30 docs	0.2507
At 100 docs	0.1588
At 200 docs	0.1200
At 500 docs	0.0690
At 1000 docs	0.0414
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2406



Recall-Precision Curve

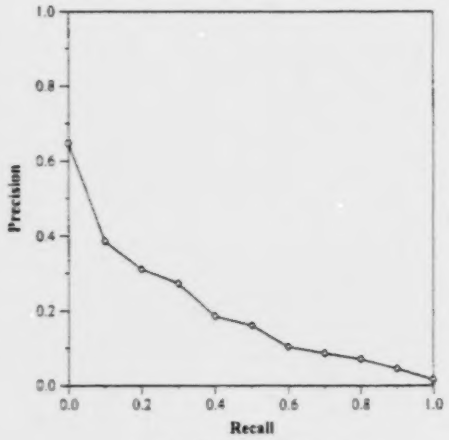


Difference from Median in Average Precision per Topic

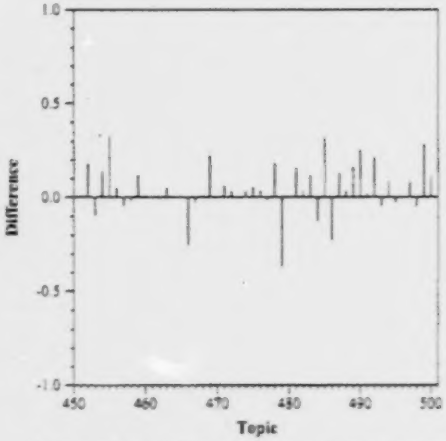
Summary Statistics	
Run Number	xvsmman
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	25171
Relevant:	2617
Rel-ret:	1363

Recall Level Precision Averages	
Recall	Precision
0.00	0.6480
0.10	0.3865
0.20	0.3111
0.30	0.2725
0.40	0.1873
0.50	0.1621
0.60	0.1039
0.70	0.0864
0.80	0.0707
0.90	0.0440
1.00	0.0154
Average precision over all relevant docs	
non-interpolated	0.1785

Document Level Averages	
	Precision
At 5 docs	0.3080
At 10 docs	0.2600
At 15 docs	0.2280
At 20 docs	0.2140
At 30 docs	0.1840
At 100 docs	0.1220
At 200 docs	0.0894
At 500 docs	0.0488
At 1000 docs	0.0273
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2211



Recall-Precision Curve

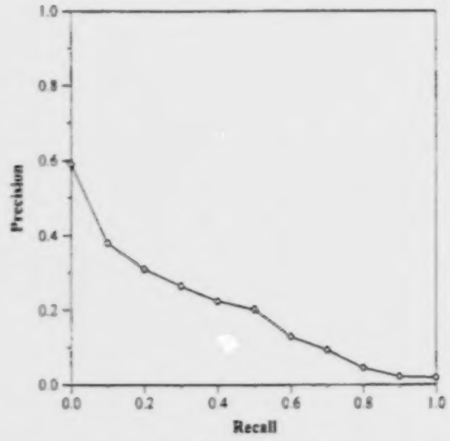


Difference from Median in Average Precision per Topic

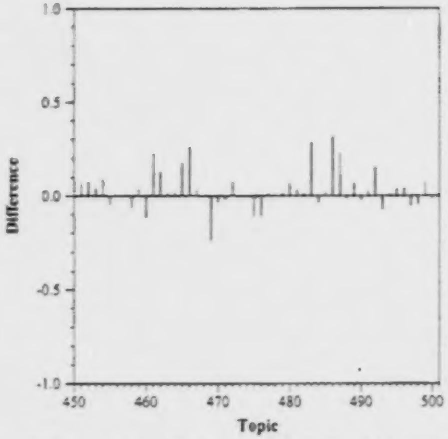
Summary Statistics	
Run Number	Flab9atd2N
Run Description	Automatic, content-only, title+desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1490

Recall Level Precision Averages	
Recall	Precision
0.00	0.5919
0.10	0.3802
0.20	0.3094
0.30	0.2646
0.40	0.2241
0.50	0.2020
0.60	0.1295
0.70	0.0936
0.80	0.0456
0.90	0.0222
1.00	0.0192
Average precision over all relevant docs	
non-interpolated	0.1877

Document Level Averages	
	Precision
At 5 docs	0.3520
At 10 docs	0.3020
At 15 docs	0.2507
At 20 docs	0.2260
At 30 docs	0.1947
At 100 docs	0.1270
At 200 docs	0.0899
At 500 docs	0.0487
At 1000 docs	0.0298
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2088



Recall-Precision Curve



Difference from Median in Average Precision per Topic

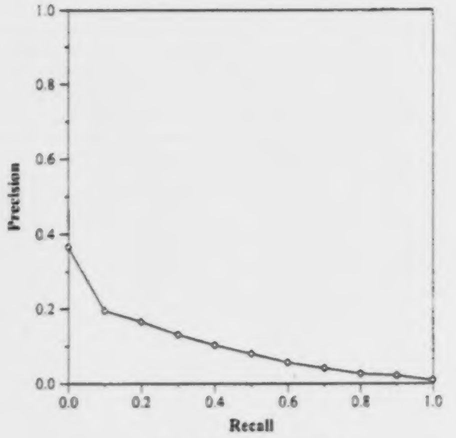
Main web track results — Seoul National University

Summary Statistics	
Run Number	Scai9Web1
Run Description	Automatic, content-only, title+desc
Number of Topics	50

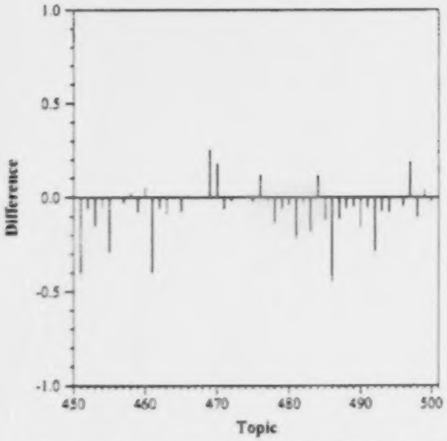
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1291

Recall Level Precision Averages	
Recall	Precision
0.00	0.3665
0.10	0.1953
0.20	0.1660
0.30	0.1312
0.40	0.1030
0.50	0.0801
0.60	0.0564
0.70	0.0414
0.80	0.0262
0.90	0.0213
1.00	0.0085
Average precision over all relevant docs	
non-interpolated	0.0941

Document Level Averages	
	Precision
At 5 docs	0.1800
At 10 docs	0.1520
At 15 docs	0.1360
At 20 docs	0.1290
At 30 docs	0.1140
At 100 docs	0.0836
At 200 docs	0.0653
At 500 docs	0.0428
At 1000 docs	0.0258
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1208



Recall-Precision Curve

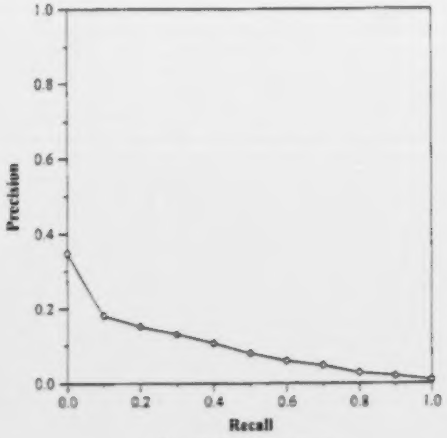


Difference from Median in Average Precision per Topic

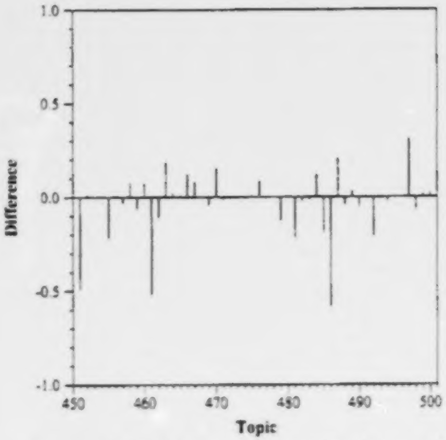
Summary Statistics	
Run Number	Scai9Web3
Run Description	Automatic, content-only, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	46549
Relevant:	2617
Rel-ret:	1289

Recall Level Precision Averages	
Recall	Precision
0.00	0.3477
0.10	0.1811
0.20	0.1519
0.30	0.1306
0.40	0.1062
0.50	0.0792
0.60	0.0587
0.70	0.0469
0.80	0.0276
0.90	0.0204
1.00	0.0090
Average precision over all relevant docs	
non-interpolated	0.0915

Document Level Averages	
	Precision
At 5 docs	0.1720
At 10 docs	0.1540
At 15 docs	0.1240
At 20 docs	0.1200
At 30 docs	0.1033
At 100 docs	0.0794
At 200 docs	0.0663
At 500 docs	0.0423
At 1000 docs	0.0258
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1126



Recall-Precision Curve



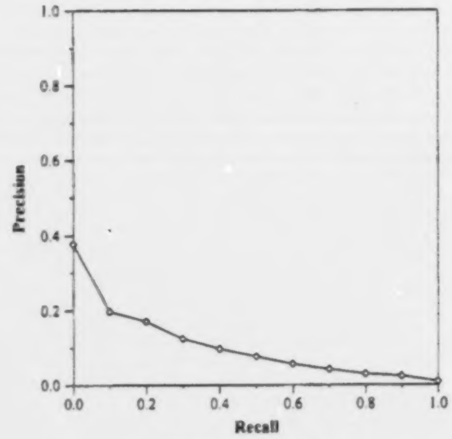
Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	Scai9Web2
Run Description	Automatic, content-only, title+desc
Number of Topics	50

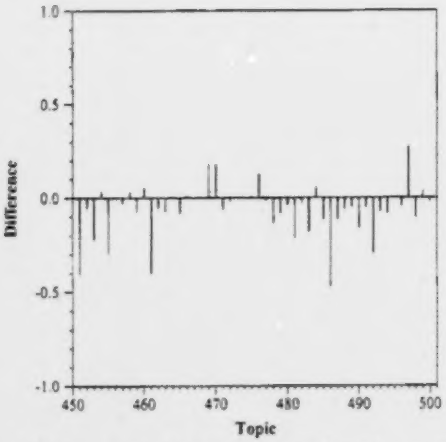
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1246

Recall Level Precision Averages	
Recall	Precision
0.00	0.3779
0.10	0.1984
0.20	0.1717
0.30	0.1244
0.40	0.0967
0.50	0.0766
0.60	0.0554
0.70	0.0410
0.80	0.0289
0.90	0.0233
1.00	0.0089
Average precision over all relevant docs	
non-interpolated	0.0934

Document Level Averages	
	Precision
At 5 docs	0.1720
At 10 docs	0.1380
At 15 docs	0.1253
At 20 docs	0.1250
At 30 docs	0.1100
At 100 docs	0.0830
At 200 docs	0.0640
At 500 docs	0.0406
At 1000 docs	0.0249
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1167



Recall-Precision Curve



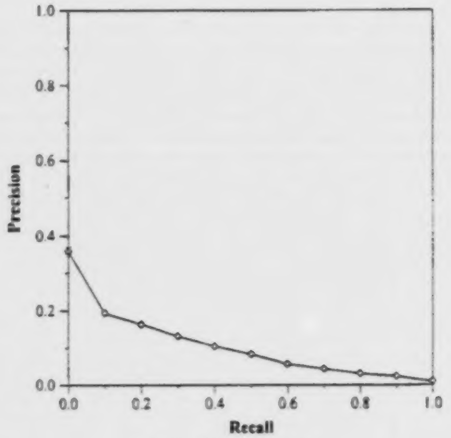
Difference from Median in Average Precision per Topic

Main web track results — Seoul National University

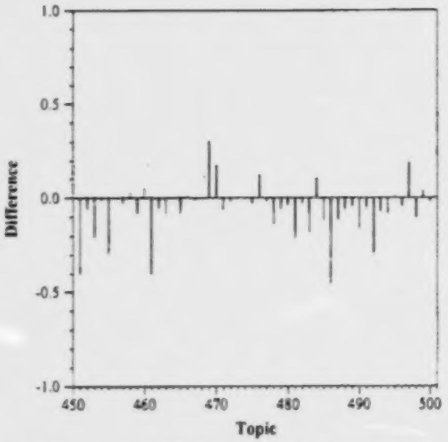
Summary Statistics	
Run Number	Scai9Web4
Run Description	Automatic, content-only, title+desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1275

Recall Level Precision Averages	
Recall	Precision
0.00	0.3603
0.10	0.1942
0.20	0.1637
0.30	0.1326
0.40	0.1053
0.50	0.0835
0.60	0.0560
0.70	0.0429
0.80	0.0299
0.90	0.0222
1.00	0.0092
Average precision over all relevant docs	
non-interpolated	0.0946

Document Level Averages	
	Precision
At 5 docs	0.1720
At 10 docs	0.1460
At 15 docs	0.1280
At 20 docs	0.1280
At 30 docs	0.1113
At 100 docs	0.0838
At 200 docs	0.0645
At 500 docs	0.0418
At 1000 docs	0.0255
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1228



Recall-Precision Curve



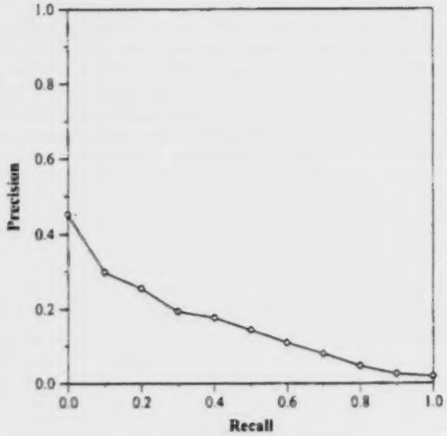
Difference from Median in Average Precision per Topic

Main web track results — AT&T Labs

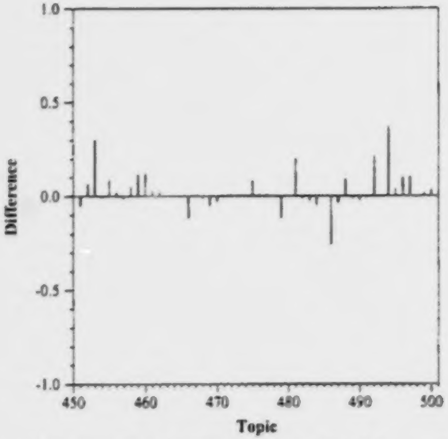
Summary Statistics	
Run Number	att0010gbe
Run Description	Automatic, content-only, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	44565
Relevant:	2617
Rel-ret:	1288

Recall Level Precision Averages	
Recall	Precision
0.00	0.4528
0.10	0.2977
0.20	0.2546
0.30	0.1934
0.40	0.1769
0.50	0.1434
0.60	0.1081
0.70	0.0781
0.80	0.0453
0.90	0.0247
1.00	0.0177
Average precision over all relevant docs	
non-interpolated	0.1464

Document Level Averages	
	Precision
At 5 docs	0.2480
At 10 docs	0.2260
At 15 docs	0.2107
At 20 docs	0.1940
At 30 docs	0.1767
At 100 docs	0.1130
At 200 docs	0.0776
At 500 docs	0.0442
At 1000 docs	0.0258
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1715



Recall-Precision Curve

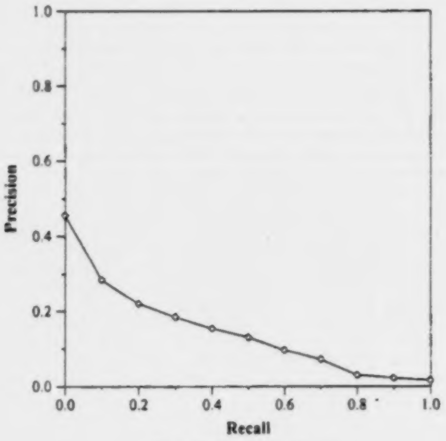


Difference from Median in Average Precision per Topic

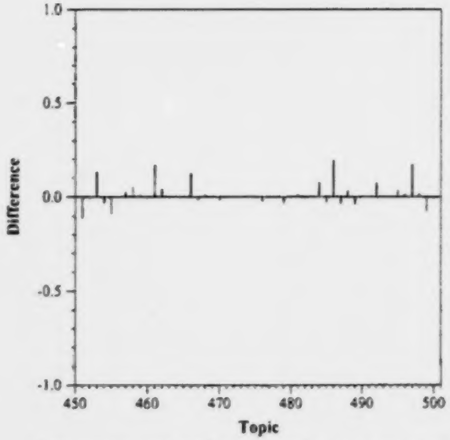
Summary Statistics	
Run Number	att0010gb
Run Description	Automatic, content-only, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	43490
Relevant:	2617
Rel-ret:	1187

Recall Level Precision Averages	
Recall	Precision
0.00	0.4558
0.10	0.2849
0.20	0.2208
0.30	0.1852
0.40	0.1539
0.50	0.1307
0.60	0.0960
0.70	0.0725
0.80	0.0302
0.90	0.0225
1.00	0.0165
Average precision over all relevant docs	
non-interpolated	0.1341

Document Level Averages	
	Precision
At 5 docs	0.2400
At 10 docs	0.2000
At 15 docs	0.1760
At 20 docs	0.1620
At 30 docs	0.1460
At 100 docs	0.0942
At 200 docs	0.0659
At 500 docs	0.0370
At 1000 docs	0.0237
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1590



Recall-Precision Curve



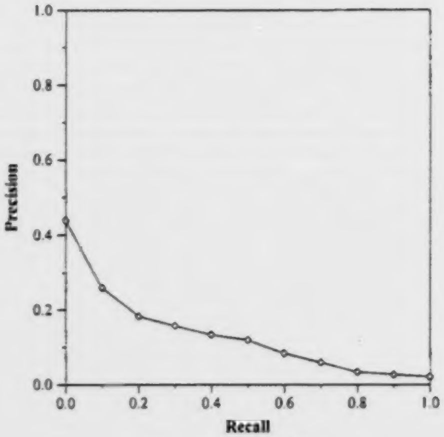
Difference from Median in Average Precision per Topic

Main web track results — AT&T Labs

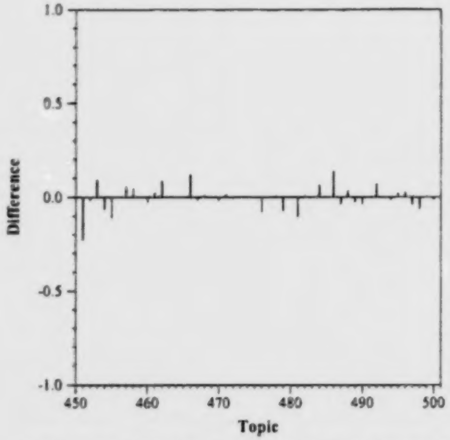
Summary Statistics	
Run Number	att0010gbt
Run Description	Automatic, content-only, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	43492
Relevant:	2617
Rel-ret:	1192

Recall Level Precision Averages	
Recall	Precision
0.00	0.4389
0.10	0.2591
0.20	0.1837
0.30	0.1581
0.40	0.1333
0.50	0.1197
0.60	0.0832
0.70	0.0597
0.80	0.0332
0.90	0.0265
1.00	0.0204
Average precision over all relevant docs	
non-interpolated	0.1182

Document Level Averages	
	Precision
At 5 docs	0.2000
At 10 docs	0.1720
At 15 docs	0.1640
At 20 docs	0.1460
At 30 docs	0.1233
At 100 docs	0.0842
At 200 docs	0.0629
At 500 docs	0.0366
At 1000 docs	0.0238
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1470



Recall-Precision Curve



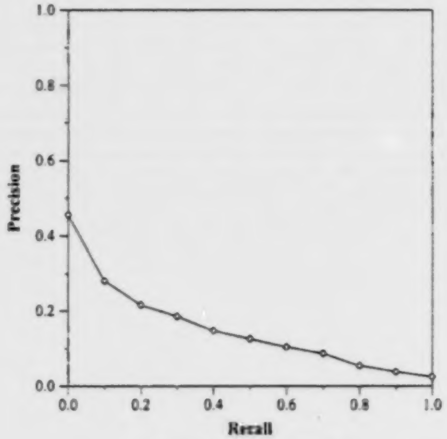
Difference from Median in Average Precision per Topic

Main web track results — AT&T Labs

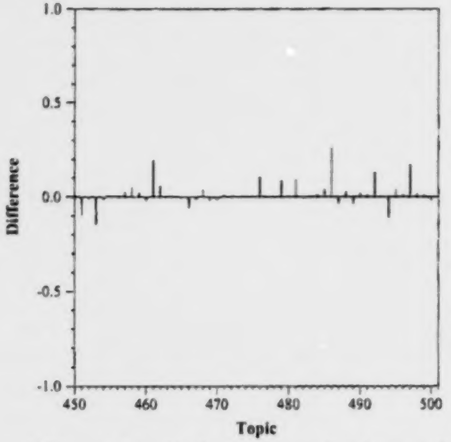
Summary Statistics	
Run Number	att0010gbl
Run Description	Automatic, content-only, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	43490
Relevant:	2617
Rel-ret:	1031

Recall Level Precision Averages	
Recall	Precision
0.00	0.4554
0.10	0.2809
0.20	0.2165
0.30	0.1862
0.40	0.1482
0.50	0.1267
0.60	0.1051
0.70	0.0876
0.80	0.0549
0.90	0.0390
1.00	0.0255
Average precision over all relevant docs	
non-interpolated	0.1380

Document Level Averages	
	Precision
At 5 docs	0.2440
At 10 docs	0.2040
At 15 docs	0.1813
At 20 docs	0.1650
At 30 docs	0.1413
At 100 docs	0.0948
At 200 docs	0.0677
At 500 docs	0.0358
At 1000 docs	0.0206
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1606



Recall-Precision Curve

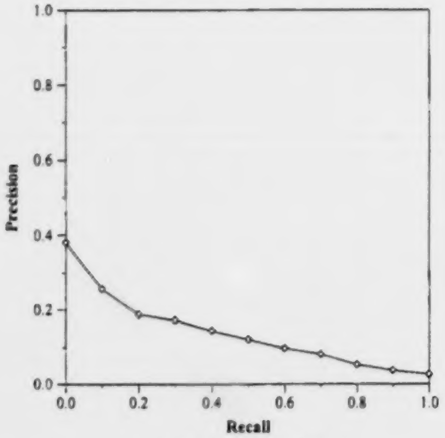


Difference from Median in Average Precision per Topic

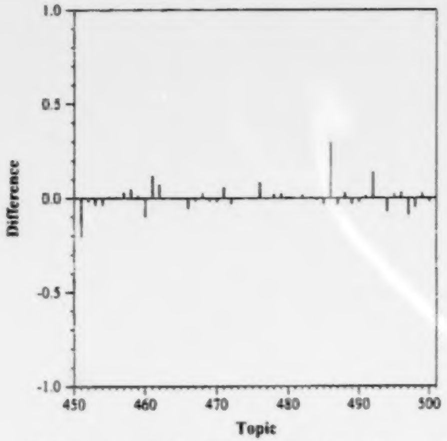
Summary Statistics	
Run Number	att0010glf
Run Description	Automatic, content-link, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	43496
Relevant:	2617
Rel-ret:	1064

Recall Level Precision Averages	
Recall	Precision
0.00	0.3809
0.10	0.2552
0.20	0.1878
0.30	0.1727
0.40	0.1442
0.50	0.1219
0.60	0.0971
0.70	0.0811
0.80	0.0525
0.90	0.0371
1.00	0.0260
Average precision over all relevant docs	
non-interpolated	0.1250

Document Level Averages	
	Precision
At 5 docs	0.1880
At 10 docs	0.1820
At 15 docs	0.1773
At 20 docs	0.1590
At 30 docs	0.1373
At 100 docs	0.0926
At 200 docs	0.0682
At 500 docs	0.0370
At 1000 docs	0.0213
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1484



Recall-Precision Curve



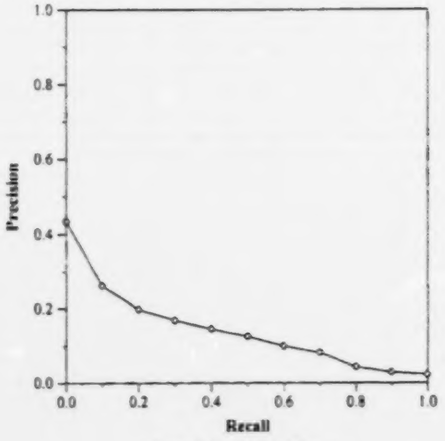
Difference from Median in Average Precision per Topic

Main web track results — AT&T Labs

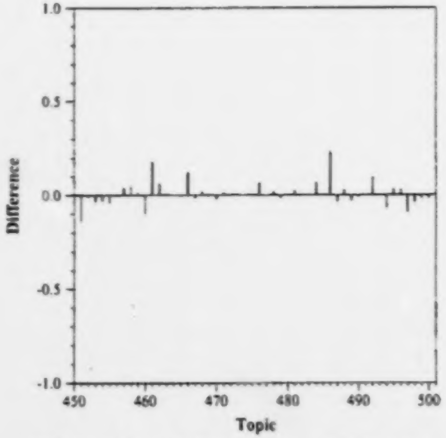
Summary Statistics	
Run Number	att0010glv
Run Description	Automatic, content-link, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	43496
Relevant:	2617
Rel-ret:	1064

Recall Level Precision Averages	
Recall	Precision
0.00	0.4347
0.10	0.2631
0.20	0.1981
0.30	0.1693
0.40	0.1464
0.50	0.1262
0.60	0.1001
0.70	0.0823
0.80	0.0422
0.90	0.0285
1.00	0.0218
Average precision over all relevant docs	
non-interpolated	0.1288

Document Level Averages	
	Precision
At 5 docs	0.2160
At 10 docs	0.1900
At 15 docs	0.1760
At 20 docs	0.1560
At 30 docs	0.1373
At 100 docs	0.0948
At 200 docs	0.0659
At 500 docs	0.0364
At 1000 docs	0.0213
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1540



Recall-Precision Curve



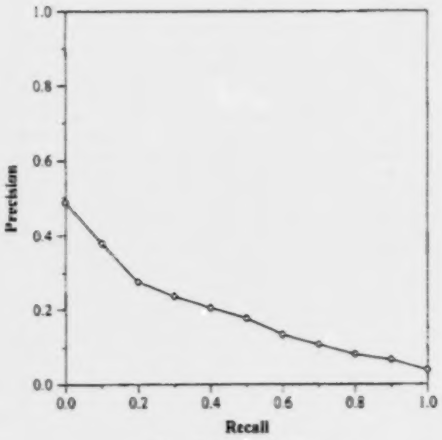
Difference from Median in Average Precision per Topic

Main web track results — University of Waterloo MultiText project

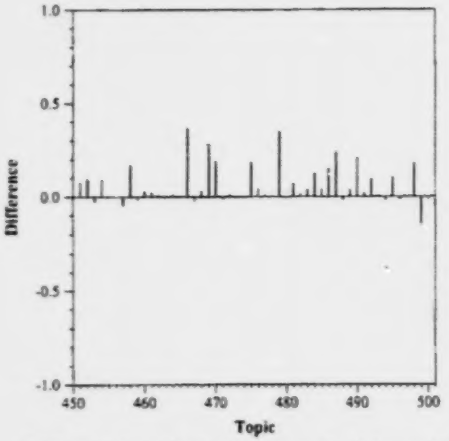
Summary Statistics	
Run Number	uwmt9w10g4
Run Description	Automatic, content-only, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49208
Relevant:	2617
Rel-ret:	1499

Recall Level Precision Averages	
Recall	Precision
0.00	0.4873
0.10	0.3802
0.20	0.2754
0.30	0.2366
0.40	0.2055
0.50	0.1784
0.60	0.1353
0.70	0.1091
0.80	0.0817
0.90	0.0679
1.00	0.0386
Average precision over all relevant docs	
non-interpolated	0.1812

Document Level Averages	
	Precision
At 5 docs	0.2760
At 10 docs	0.2400
At 15 docs	0.2107
At 20 docs	0.1880
At 30 docs	0.1740
At 100 docs	0.1152
At 200 docs	0.0891
At 500 docs	0.0518
At 1000 docs	0.0300
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1937



Recall-Precision Curve



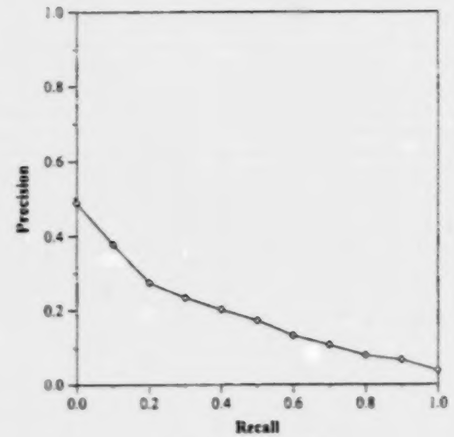
Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	uwmt9w10g5
Run Description	Automatic, content-link, title only
Number of Topics	50

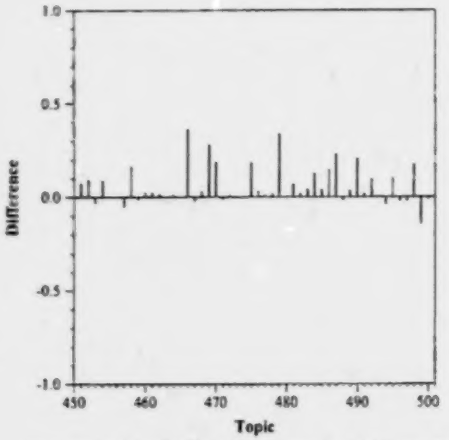
Total number of documents over all topics	
Retrieved:	49208
Relevant:	2617
Rel-ret:	1499

Recall Level Precision Averages	
Recall	Precision
0.00	0.4883
0.10	0.3789
0.20	0.2747
0.30	0.2341
0.40	0.2035
0.50	0.1746
0.60	0.1339
0.70	0.1085
0.80	0.0799
0.90	0.0672
1.00	0.0381
Average precision over all relevant docs	
non-interpolated	0.1794

Document Level Averages	
	Precision
At 5 docs	0.2760
At 10 docs	0.2400
At 15 docs	0.2107
At 20 docs	0.1860
At 30 docs	0.1653
At 100 docs	0.1158
At 200 docs	0.0881
At 500 docs	0.0518
At 1000 docs	0.0300
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1936



Recall-Precision Curve



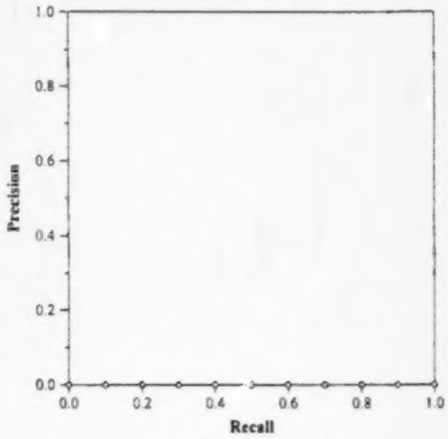
Difference from Median in Average Precision per Topic

Main web track results — Pam Wood

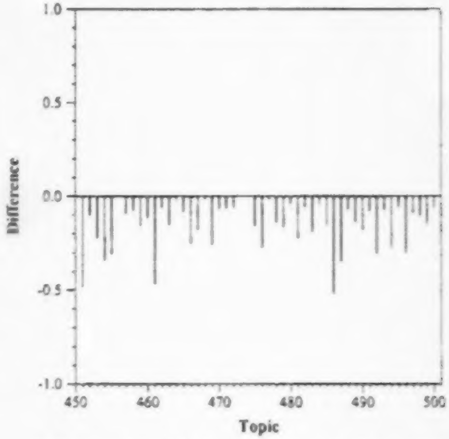
Summary Statistics	
Run Number	UCCS3
Run Description	Automatic, content-only, description
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	3

Recall Level Precision Averages	
Recall	Precision
0.00	0.0001
0.10	0.0000
0.20	0.0000
0.30	0.0000
0.40	0.0000
0.50	0.0000
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0000

Document Level Averages	
	Precision
At 5 docs	0.0000
At 10 docs	0.0000
At 15 docs	0.0000
At 20 docs	0.0000
At 30 docs	0.0000
At 100 docs	0.0000
At 200 docs	0.0000
At 500 docs	0.0000
At 1000 docs	0.0001
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0000



Recall-Precision Curve

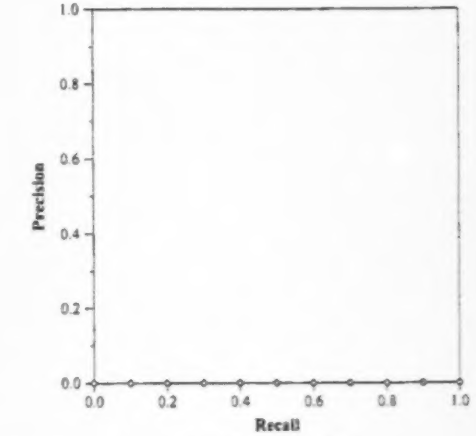


Difference from Median in Average Precision per Topic

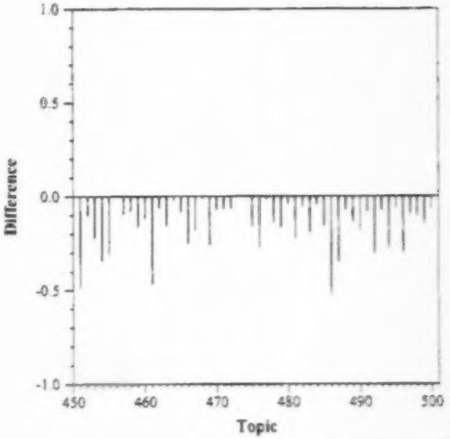
Summary Statistics	
Run Number	UCCS4
Run Description	Automatic, content-only, description
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	5

Recall Level Precision Averages	
Recall	Precision
0.00	0.0002
0.10	0.0000
0.20	0.0000
0.30	0.0000
0.40	0.0000
0.50	0.0000
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0000

Document Level Averages	
	Precision
At 5 docs	0.0000
At 10 docs	0.0000
At 15 docs	0.0000
At 20 docs	0.0000
At 30 docs	0.0000
At 100 docs	0.0000
At 200 docs	0.0000
At 500 docs	0.0000
At 1000 docs	0.0001
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0000



Recall-Precision Curve

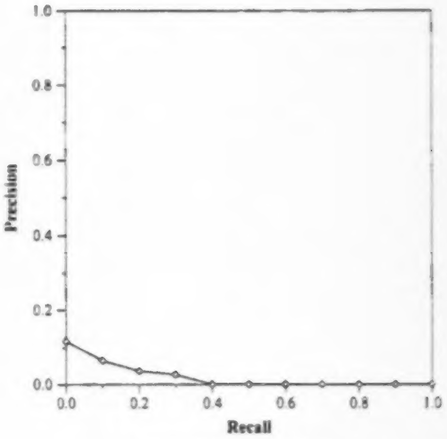


Difference from Median in Average Precision per Topic

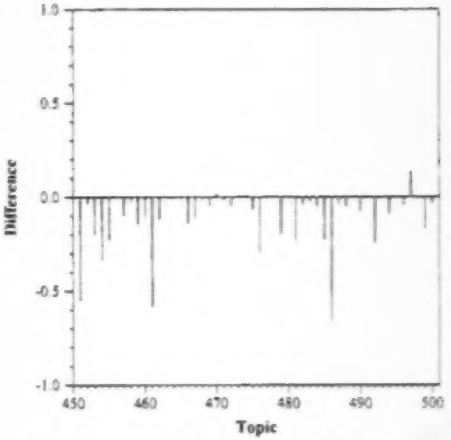
Summary Statistics	
Run Number	UCCS1
Run Description	Automatic, content-only, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	41950
Relevant:	2617
Rel-ret:	173

Recall Level Precision Averages	
Recall	Precision
0.00	0.1184
0.10	0.0651
0.20	0.0366
0.30	0.0260
0.40	0.0003
0.50	0.0003
0.60	0.0003
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0181

Document Level Averages	
	Precision
At 5 docs	0.0640
At 10 docs	0.0520
At 15 docs	0.0400
At 20 docs	0.0340
At 30 docs	0.0267
At 100 docs	0.0170
At 200 docs	0.0102
At 500 docs	0.0059
At 1000 docs	0.0035
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0314



Recall-Precision Curve



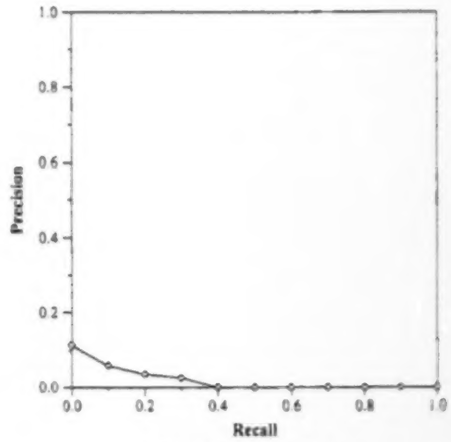
Difference from Median in Average Precision per Topic

Main web track results — Pam Wood

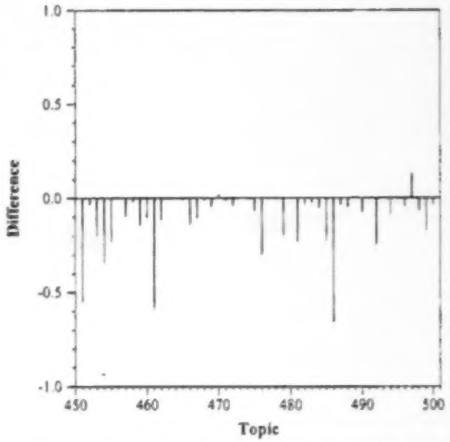
Summary Statistics	
Run Number	UCCS2
Run Description	Automatic, content-only, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	41950
Relevant:	2617
Rel-ret:	173

Recall Level Precision Averages	
Recall	Precision
0.00	0.1141
0.10	0.0601
0.20	0.0361
0.30	0.0257
0.40	0.0000
0.50	0.0000
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0169

Document Level Averages	
	Precision
At 5 docs	0.0600
At 10 docs	0.0520
At 15 docs	0.0387
At 20 docs	0.0320
At 30 docs	0.0267
At 100 docs	0.0162
At 200 docs	0.0103
At 500 docs	0.0058
At 1000 docs	0.0035
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0290



Recall-Precision Curve



Difference from Median in Average Precision per Topic

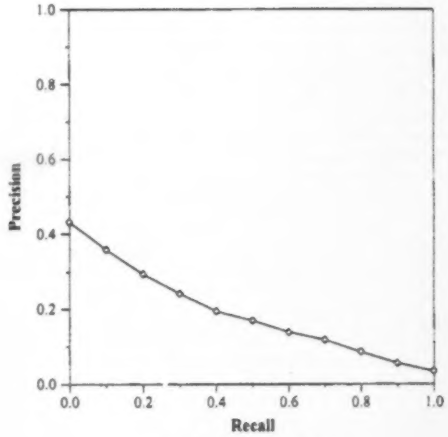
Main web track results — RICOH Co., Ltd.

Summary Statistics	
Run Number	ric9tpx
Run Description	Automatic, content-only, title only
Number of Topics	50

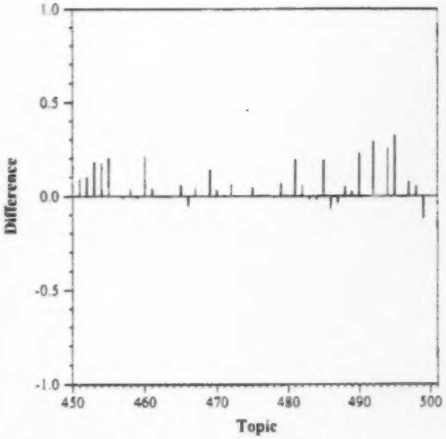
Total number of documents over all topics	
Retrieved:	48283
Relevant:	2617
Rel-ret:	1685

Recall Level Precision Averages	
Recall	Precision
0.00	0.4326
0.10	0.3587
0.20	0.2951
0.30	0.2426
0.40	0.1944
0.50	0.1690
0.60	0.1377
0.70	0.1166
0.80	0.0864
0.90	0.0568
1.00	0.0354
Average precision over all relevant docs	
non-interpolated	0.1787

Document Level Averages	
	Precision
At 5 docs	0.3000
At 10 docs	0.2760
At 15 docs	0.2413
At 20 docs	0.2290
At 30 docs	0.2080
At 100 docs	0.1324
At 200 docs	0.0939
At 500 docs	0.0563
At 1000 docs	0.0337
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1864



Recall-Precision Curve



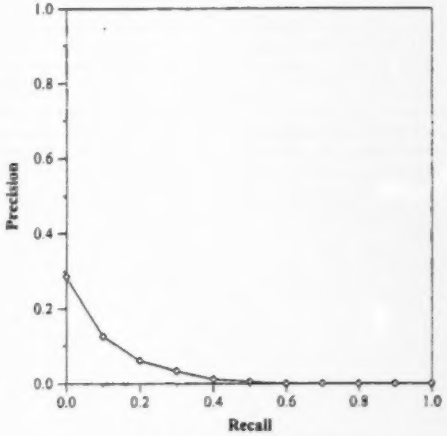
Difference from Median in Average Precision per Topic

Main web track results — University of North Carolina at Chapel Hill

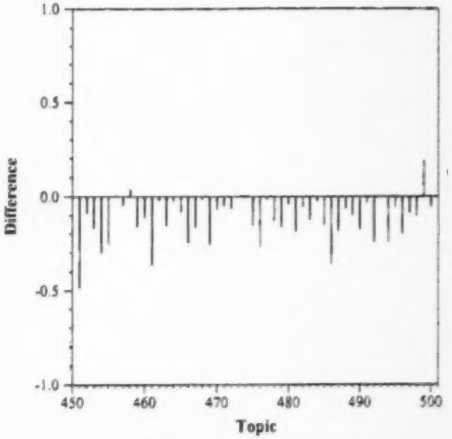
Summary Statistics	
Run Number	iswtd
Run Description	Automatic, content-only, title+desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	236

Recall Level Precision Averages	
Recall	Precision
0.00	0.2860
0.10	0.1251
0.20	0.0617
0.30	0.0338
0.40	0.0117
0.50	0.0051
0.60	0.0001
0.70	0.0001
0.80	0.0001
0.90	0.0001
1.00	0.0001
Average precision over all relevant docs	
non-interpolated	0.0325

Document Level Averages	
	Precision
At 5 docs	0.1440
At 10 docs	0.1100
At 15 docs	0.0893
At 20 docs	0.0730
At 30 docs	0.0593
At 100 docs	0.0260
At 200 docs	0.0149
At 500 docs	0.0082
At 1000 docs	0.0047
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0618



Recall-Precision Curve



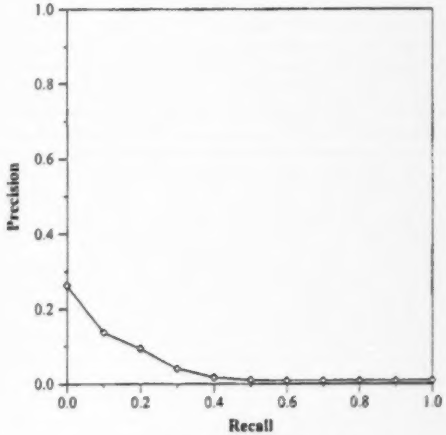
Difference from Median in Average Precision per Topic

Main web track results — University of North Carolina at Chapel Hill

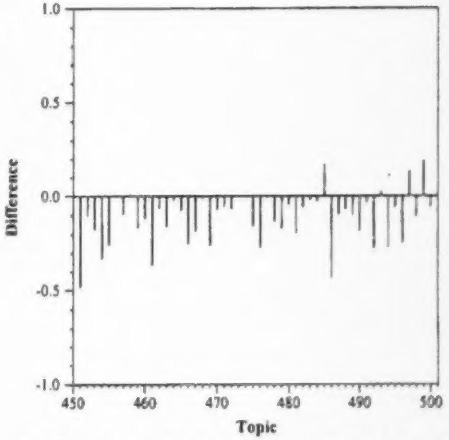
Summary Statistics	
Run Number	iswtdn
Run Description	Automatic, content-only, title+desc+narr
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	172

Recall Level Precision Averages	
Recall	Precision
0.00	0.2634
0.10	0.1377
0.20	0.0939
0.30	0.0403
0.40	0.0174
0.50	0.0099
0.60	0.0080
0.70	0.0080
0.80	0.0080
0.90	0.0080
1.00	0.0080
Average precision over all relevant docs	
non-interpolated	0.0412

Document Level Averages	
	Precision
At 5 docs	0.1200
At 10 docs	0.0840
At 15 docs	0.0667
At 20 docs	0.0520
At 30 docs	0.0427
At 100 docs	0.0192
At 200 docs	0.0107
At 500 docs	0.0057
At 1000 docs	0.0034
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0580



Recall-Precision Curve



Difference from Median in Average Precision per Topic

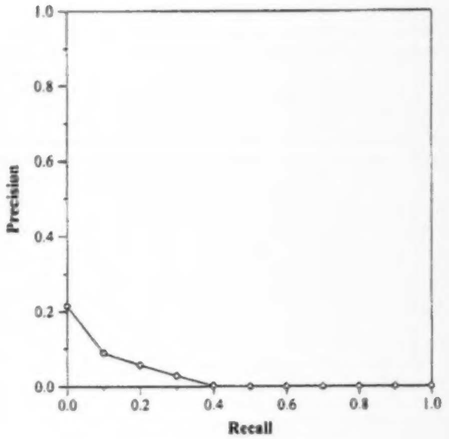
Main web track results — University of North Carolina at Chapel Hill

Summary Statistics	
Run Number	iswt
Run Description	Automatic, content-only, title only
Number of Topics	50

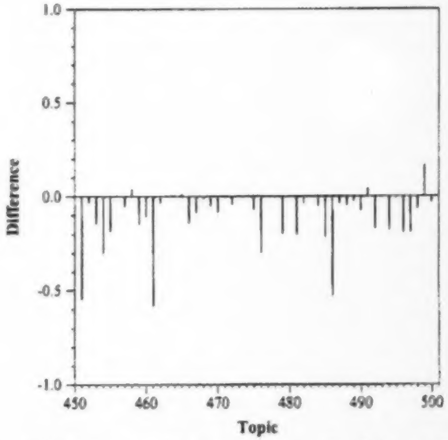
Total number of documents over all topics	
Retrieved:	39823
Relevant:	2617
Rel-ret:	242

Recall Level Precision Averages	
Recall	Precision
0.00	0.2153
0.10	0.0891
0.20	0.0566
0.30	0.0289
0.40	0.0021
0.50	0.0004
0.60	0.0003
0.70	0.0003
0.80	0.0003
0.90	0.0003
1.00	0.0003
Average precision over all relevant docs	
non-interpolated	0.0240

Document Level Averages	
	Precision
At 5 docs	0.0880
At 10 docs	0.0760
At 15 docs	0.0573
At 20 docs	0.0530
At 30 docs	0.0440
At 100 docs	0.0244
At 200 docs	0.0156
At 500 docs	0.0085
At 1000 docs	0.0048
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0484



Recall-Precision Curve



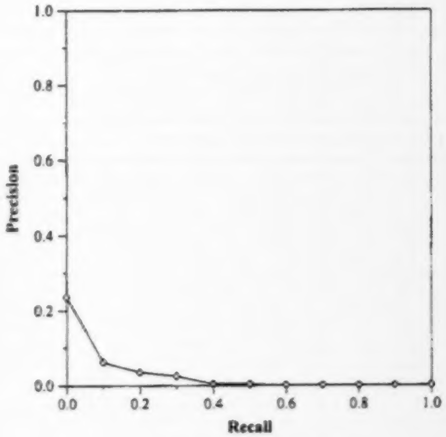
Difference from Median in Average Precision per Topic

Main web track results — University of North Carolina at Chapel Hill

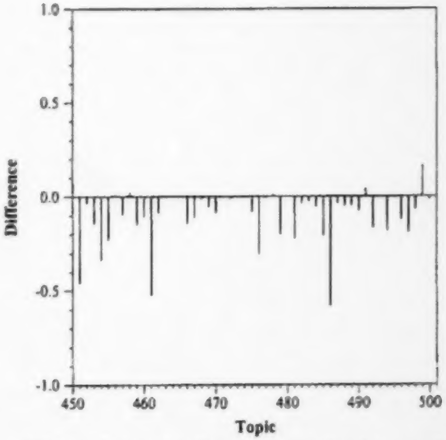
Summary Statistics	
Run Number	isnnwt
Run Description	Automatic, content-only, title only
Number of Topics	50
Total number of documents over all topics	
Retrieved:	46811
Relevant:	2617
Rel-ret:	126

Recall Level Precision Averages	
Recall	Precision
0.00	0.2372
0.10	0.0618
0.20	0.0357
0.30	0.0255
0.40	0.0044
0.50	0.0041
0.60	0.0006
0.70	0.0006
0.80	0.0006
0.90	0.0006
1.00	0.0006
Average precision over all relevant docs	
non-interpolated	0.0225

Document Level Averages	
	Precision
At 5 docs	0.0880
At 10 docs	0.0580
At 15 docs	0.0440
At 20 docs	0.0390
At 30 docs	0.0300
At 100 docs	0.0150
At 200 docs	0.0096
At 500 docs	0.0045
At 1000 docs	0.0025
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0366



Recall-Precision Curve



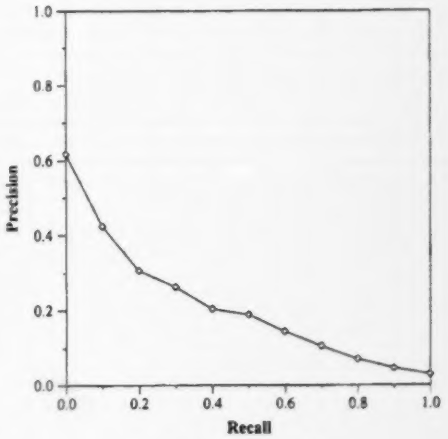
Difference from Median in Average Precision per Topic

Main web track results — Johns Hopkins University, APL

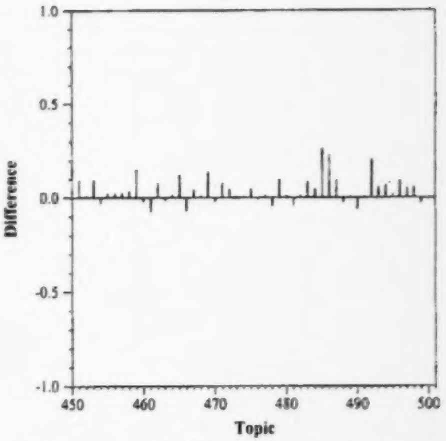
Summary Statistics	
Run Number	apl9all
Run Description	Automatic, content-only, title+desc+narr
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1609

Recall Level Precision Averages	
Recall	Precision
0.00	0.6178
0.10	0.4239
0.20	0.3073
0.30	0.2634
0.40	0.2046
0.50	0.1890
0.60	0.1446
0.70	0.1066
0.80	0.0712
0.90	0.0465
1.00	0.0291
Average precision over all relevant docs	
non-interpolated	0.1948

Document Level Averages	
	Precision
At 5 docs	0.3680
At 10 docs	0.3140
At 15 docs	0.2773
At 20 docs	0.2440
At 30 docs	0.2093
At 100 docs	0.1320
At 200 docs	0.0994
At 500 docs	0.0552
At 1000 docs	0.0322
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2111



Recall-Precision Curve



Difference from Median in Average Precision per Topic

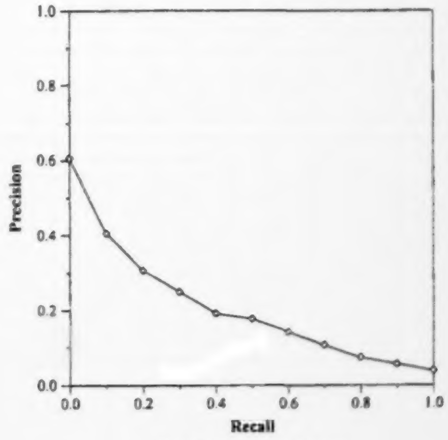
Main web track results — Johns Hopkins University, APL

Summary Statistics	
Run Number	apl9td
Run Description	Automatic, content-only, title+desc
Number of Topics	50

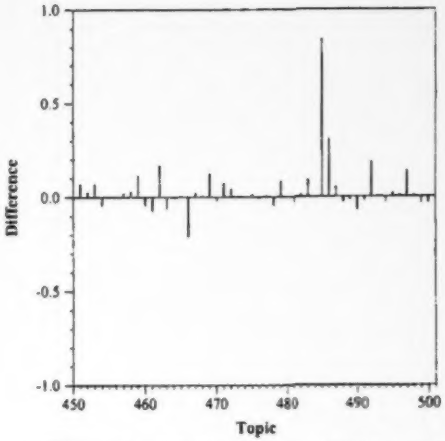
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2617
Rel-ret:	1535

Recall Level Precision Averages	
Recall	Precision
0.00	0.6057
0.10	0.4046
0.20	0.3062
0.30	0.2510
0.40	0.1920
0.50	0.1764
0.60	0.1414
0.70	0.1057
0.80	0.0733
0.90	0.0552
1.00	0.0385
Average precision over all relevant docs	
non-interpolated	0.1917

Document Level Averages	
	Precision
At 5 docs	0.3400
At 10 docs	0.2940
At 15 docs	0.2480
At 20 docs	0.2240
At 30 docs	0.1940
At 100 docs	0.1248
At 200 docs	0.0933
At 500 docs	0.0516
At 1000 docs	0.0307
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2127



Recall-Precision Curve



Difference from Median in Average Precision per Topic

U.S. Department of Education
Office of Educatonal Research and Improvement (OERI)
National Library of Education (NLE)
Educational Resources Information Center (ERIC)

ERIC®

REPRODUCTION RELEASE
(Specific Document)

I. DOCUMENT IDENTIFICATION:

Title:	Proceedings of the Ninth Text REtrieval Conference (TREC-9)		
Authors:	Ellen Voorhees, Ed.		
Corporate Source:	National Institute of Standards and Technology (NIST)	Publication Date:	2000

II. REPRODUCTION RELEASE:

In order to disseminate as widely as possible timely and significant materials of interest to the educational community, documents announced in the monthly abstract journal of the ERIC system, Resources in Education (RIE), are usually made available to users in microfiche, reproduced paper copy, and electronic media, and sold through the ERIC Document Reproduction Service (EDRS). Credit is given to the source of each document, and, if reproduction release is granted, one of the following notices is affixed to the document.

If permission is granted to reproduce and disseminate the identified document, please CHECK ONE of the following three options below and sign at the bottom of the page.

The sample sticker shown below will be affixed to all Level 1 documents	he sample sticker shown below will be affixed to all Level 2A documents	The sample sticker shown below will be affixed to all Level 2B documents
PERMISSION TO REPRODUCE AND DISSEMINATE THIS MATERIAL HAS BEEN GRANTED BY SAMPLE TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)	PERMISSION TO REPRODUCE AND DISSEMINATE THIS MATERIAL IN MICROFICHE, AND IN ELECTRONIC MEDIA FOR ERIC COLLECTION SUBSCRIBERS ONLY HAS BEEN GRANTED BY SAMPLE TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)	PERMISSION TO REPRODUCE AND DISSEMINATE THIS MATERIAL IN MICROFICHE ONLY HAS BEEN GRANTED BY SAMPLE TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
Level 1	Level 2A	Level 2B
●	○	○
Check here for Level 1 release, permitting reproduction and dissemination in microfiche or other ERIC archival media (e.g., electronic) and paper copy.	Check here for Level 2A release, permitting reproduction and dissemination in microfiche and in electronic media for ERIC archival collection subscribers only.	Check here for Level 2B release, permitting reproduction and dissemination in microfiche only.
Documents will be processed as indicated provided reproduction quality permits. If permission to reproduce is granted, but no box is checked, documents will be processed at Level 1.		

I hereby grant to the Educational Resources Information Center (ERIC) nonexclusive permission to reproduce and disseminate this document as indicated above. Reproducation from the ERIC microfiche or electronic media by persons other than ERIC employees and its system contractors requires permission from the copyright holder. Exception is made for non-profit reproduction by libraries and other service agencies to satisfy information needs of educators in response to discrete inquiries.

Signature: Ellen M Voorhees	Printed Name/Position/Title: Ellen Voorhees, TREC Project Manager	
Organization/Address: National Institute of Standards and Technology (NIST)/100 Bureau Drive/Gaithersburg, MD 20899-8940	Telephone: 301-975-3761	FAX: 301-975-5287
	E-mail Address: ellen.voorhees@nist.gov	Date: January 29, 2003

EFF-088 (Rev. 2/2001)

III. DOCUMENT AVAILABILITY INFORMATION (FROM NON-ERIC SOURCE):

If permission to reproduce is not granted to ERIC, *or*, if you wish ERIC to cite the availability of these documents from another source, please provide the following information regarding the availability of these documents. (ERIC will not announce a document unless it is publicly available, and a dependable source can be specified. Contributors should also be aware that ERIC selection criteria are significantly more stringent for documents that cannot be made available through EDRS.)

Publisher/Distributor:
Address:
Price:

IV. REFERRAL OF ERIC TO COPYRIGHT/REPRODUCTION RIGHTS HOLDER:

If the right to grant this reproduction release is held by someone other than the addressee, please provide the appropriate name and address:

Name:
Address:

V. WHERE TO SEND THIS FORM:

Send this form to the following ERIC Clearinghouse:

However, if solicited by the ERIC Facility, or if making an unsolicited contribution to ERIC, return this form (and the documents being contributed) to:

ERIC Processing and Reference Facility
4483-A Forbes Boulevard
Lanham, Maryland 20706

Telephone: 301-552-4200
Toll Free: 800-799-3742
FAX: 301-552-4700
e-mail: ericfac@inet.ed.gov
WWW: http://ericfac.piccard.csc.com

EFF-087 (Rev. 2/2000)